www.GossamerSec.com

Assurance Activity Report for Cisco Adaptive Security Appliances (ASA) 9.20 on Firepower 2100 Series

Version 0.3
November 7, 2024

Prepared by:
Gossamer Security Solutions
Accredited Security Testing Laboratory – Common Criteria Testing
Columbia, MD 21045

Prepared for:
National Information Assurance Partnership
Common Criteria Evaluation and Validation Scheme

Document: AAR-VID11494

Revision History

Revision	Date	Authors	Summary
Version 0.1	September 25, 2024	Gossamer	Initial draft
Version 0.2	October 29, 2024	Gossamer	ECR Responses
Version 0.3	November 7, 2024	Gossamer	ECR Responses

The TOE Evaluation was Sponsored by:

Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134

Evaluation Personnel:

- Cornelius J. Haley
- Ryan Hagedorn
- William Micknick

Common Criteria Versions:

- Common Criteria for Information Technology Security Evaluation Part 1: Introduction, Version 3.1, Revision 5, April 2017
- Common Criteria for Information Technology Security Evaluation Part 2: Security functional components, Version 3.1, Revision 5, April 2017
- Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, Version 3.1, Revision 5, April 2017

Common Evaluation Methodology Versions:

- Common Methodology for Information Technology Security Evaluation, Evaluation Methodology, Version 3.1, Revision 5, April 2017

Table of Contents

1. Introduction

This document presents evaluations results of the Cisco Adaptive Security Appliances (ASA) 9.20 on Firepower 2100 Series NDcPP22e/STFFW14e/VPNGW13 evaluation. This document contains a description of the assurance activities and associated results as performed by the evaluators.

1.1 Equivalence

This section provides equivalence arguments for the Common Criteria testing as well as Cryptographic CAVP testing.

1.1.1 Evaluated Platform Equivalence

The TOE is the Cisco Adaptive Security Appliances (ASA) 9.20 on Firepower 2100 Series platforms. The TOE includes the following models:

Model	Processor ID	Microarchitecture
FP2110	Intel Xeon D-1526	Broadwell
FP2120	Intel Xeon D-1528	Broadwell
FP2130	Intel Xeon D-1548	Broadwell
FP2140	Intel Xeon D-1577	Broadwell

Table 1-1 2100 Series Processors

The TOE also uses a cryptographic accelerator to support the IPsec implementation. The evaluated models use the following cryptographic accelerators.

Model	Processor ID	CAVP Testing Rationale
FP2110	OCTEON III CN7230 MIPS64	CAVP Tested
FP2120	OCTEON III CN7340 MIPS64	CAVP Tested
FP2130	OCTEON III CN7350 MIPS64	CAVP Tested
FP2140	OCTEON III CN7360 MIPS64	CAVP Tested

Table 1-2 Evaluated Model Crypto Accelerator

All models are running ASA 9.20 software with the ASDM software for management. There is only one software images for ASA 9.20 running on these models. All models operate in what is called 'appliance mode', which indicates that only the ASA interfaces are exposed by the device.

All security functions provided by the TOE are implemented in software (with the exception of entropy generation and some cryptographic acceleration on Firepower 2100 series devices). The TOE security behavior is the same on all devices for each of the SFRs defined by the Security Target. These SFRs are instantiated by the same version of the TOE software and in the same way on every platform.

The ASA software image that runs on all devices was fully tested during the evaluation on the device highlighted in the table above.

1.1.2 CAVP Certificate Justification

The TOE is the Cisco Adaptive Security Appliances (ASA) 2100 Series running ASA Version 9.20 which includes a cryptographic library and a hardware cryptographic accelerator for the IPsec implementation that performs all cryptographic operations.

- CiscoSSL FOM Cryptographic implementation version 7.3a
- Octeon III family crypto engine

These cryptographic libraries perform all cryptographic operations that provide the cryptographic functions identified in the following table.

The TOE IPsec implementation uses the Octeon III family crypto engine to provide the following cryptographic functions:

- Encryption/decryption using AES in CBC or GCM modes,
- Hashing,
- Keyed Hashing, and
- RSA Signature Services.

The TOE IPsec implementation uses the FOM v7.3a to provide the following cryptographic functions:

- ECDSA signature services,
- Key establishment,
- Key generation and verification and
- Random Number Generation.

The evaluation encompasses several physical devices. These devices utilize several processors and IPsec hardware accelerators. The tables in the previous section lists the appliances, cryptographic modules and processors that are claimed in the ST. The following tables map the Cryptographic operations, related Security Functional Requirements (SFR) and standards to the CAVP certificates that demonstrate compliance to these SFR and standards.

Table 1-3 CiscoSSL FOM Version 7.3a

Functions	Requirement	Standard	Certificate #
Encryption/Decryption			
AES CBC (128 and 256 bits)	FCS_COP.1/DataEncryption	FIPS Pub 197 ISO 10116 NIST SP 800-38A ISO 19772	A 4446
AES GCM (128 and 256 bits)	FCS_COP.1/DataEncryption	ISO 19772 FIPS Pub 197 NIST SP 800-38A	A 4446
Cryptographic hashing			
SHA-1, SHA-256, SHA-384, SHA-512	FCS_COP.1/Hash	FIPS Pub 180-4 ISO/IEC 10118-3:2004	A 4446
Keyed-hash message authentication			

HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512 (digest sizes and block sizes of 160, 256, 384 and 512 bits)	FCS_COP.1/KeyedHash	FIPS Pub 198-1 FIPS Pub 180-4 ISO/IEC 9797-2:2011	A 4446
Cryptographic signature services			
RSA Digital Signature (rDSA) (2048, 3072 bits)	FCS_COP.1/SigGen	FIPS Pub 186-4 ISO/IEC 9796-2	A 4446
ECDSA Digital Signature (P-256, P-384, P-521)	FCS_COP.1/SigGen	FIPS Pub 186-4 ISO/IEC 14888-3	A 4446
Random bit generation			
HMAC_DRBG(AES) with platform based noise sources with a minimum of 256 bits of non-determinism	FCS_RBG_EXT.1	FIPS SP 800-90A ISO/IEC 18031:2011	A 4446
Key generation			
RSA Key Generation (2048-bit, 3072-bit)	FCS_CKM.1	FIPS Pub 186-4 ISO/IEC 9796-2	A 4446
ECC Key Generation (P-256, P-384, P-521)	FCS_CKM.1	FIPS PUB 186-4	A 4446
FFC Scheme using key sies of 2048-bit or greater DSA KeyPairGen	FCS_CKM.1	FIPS PUB 186-4	A 4446
FFC Schemes using 'safe-prime'	FCS_CKM.1	NIST SP 800-56A Revision 3	Tested with known good implementation
Key establishment			
RSA	FCS_CKM.2	RSAES-PKCS1-v1_5	Tested with known good implementation
KAS ECC P-256, P-384, P-521	FCS_CKM.2	NIST SP 800-56A Rev 3	A 4446
KAS FFC	FCS_CKM.2	NIST SP 800-56A Rev 3	A 4446
FFC Schemes using 'safe-prime' groups	FCS_CKM.2	NIST SP 800-56A Rev 3	Tested with known good implementation

Table 1-4 Octeon III Family Crypto Engine

Functions	Requirement	Standard	Certificate #
Encryption/Decryption			
AES CBC (128 and 256 bits)	FCS_COP.1/DataEncryption	FIPS Pub 197 ISO 10116 NIST SP 800-38A ISO 19772	AES 3301
AES GCM (128 and 256 bits)	FCS_COP.1/DataEncryption	ISO 19772 FIPS Pub 197 NIST SP 800-38A	AES 3301
Cryptographic signature services			
RSA Digital Signature (rDSA) (2048, 3072 bits)	FCS_COP.1/SigGen	FIPS Pub 186-4 ISO/IEC 9796-2	RSA 1745

Cryptographic hashing			
SHA-1, SHA-256, SHA-384, SHA-512	FCS_COP.1/Hash	FIPS Pub 180-4 ISO/IEC 10118-3:2004	SHS 2737
Keyed-hash message authentication			
HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512 (digest sizes and block sizes of 160, 256, 384 and 512 bits)	FCS_COP.1/KeyedHash	FIPS Pub 198-1 FIPS Pub 180-4 ISO/IEC 9797-2:2011	HMAC 2095

1.2 References

The following evidence was used to complete the Assurance Activities:

[**ST**]	Cisco ASA 9.20 on Firepower 2100 Series Security Target
[**AdminGuide**]	Cisco ASA 9.20 on Firepower 1000 and 2100 Series, Preparative Procedures & Operational User Guide for the Common Criteria Certified Configuration

2. Protection Profile SFR Assurance Activities

This section of the AAR identifies each of the assurance activities included in the claimed Protection Profiles and describes the findings in each case.

2.1 Security audit (FAU)

2.1.1 Audit Data Generation (NDcPP22e:FAU_GEN.1)

2.1.1.1 NDcPP22e:FAU_GEN.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.1.2 NDcPP22e:FAU_GEN.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: For the administrative task of generating/import of, changing, or deleting of cryptographic keys as defined in FAU_GEN.1.1c, the TSS should identify what information is logged to identify the relevant key.

For distributed TOEs the evaluator shall examine the TSS to ensure that it describes which of the overall required auditable events defined in FAU_GEN.1.1 are generated and recorded by which TOE components. The evaluator shall ensure that this mapping of audit events to TOE components accounts for, and is consistent with, information provided in Table 1, as well as events in Tables 2, 4, and 5 (where applicable to the overall TOE). This includes that the evaluator shall confirm that all components defined as generating audit information for a particular SFR should also contribute to that SFR as defined in the mapping of SFRs to TOE components, and that the audit records generated by each component cover all the SFRs that it implements.

Section 6.1, Table 19 (FAU_GEN.1) of [ST] provides a sample showing the type of information in a typical audit record. It also indicates that for the administrative task of generating/import of, changing, or deleting of cryptographic keys a unique key name is included in the audit record.

The TOE is not distributed.

Component Guidance Assurance Activities: The evaluator shall check the guidance documentation and ensure that it provides an example of each auditable event required by FAU_GEN.1 (i.e. at least one instance of each

auditable event, comprising the mandatory, optional and selection-based SFR sections as applicable, shall be provided from the actual audit record).

The evaluator shall also make a determination of the administrative actions related to TSF data related to configuration changes. The evaluator shall examine the guidance documentation and make a determination of which administrative commands, including subcommands, scripts, and configuration files, are related to the configuration (including enabling or disabling) of the mechanisms implemented in the TOE that are necessary to enforce the requirements specified in the cPP. The evaluator shall document the methodology or approach taken while determining which actions in the administrative guide are related to TSF data related to configuration changes. The evaluator may perform this activity as part of the activities associated with ensuring that the corresponding guidance documentation satisfies the requirements related to it.

Section "Auditable Events Certified Under Common Criteria" in the [AdminGuide] provides an example of each auditable event required by FAU_GEN.1. Section "Timestamps in Audit Messages" in the [AdminGuide] describes how the logging of timestamps must be enabled in the TOE. With “logging timestamp” enabled, all audit messages will include at least the following details: Date and time of the event, type of event, subject identity, and the outcome (success or failure) of the event. During testing, the evaluator mapped the audit event examples in the [AdminGuide] to the TOE generated events and confirmed that the [AdminGuide] includes examples/descriptions of all required audit events.

The evaluator verified the administrative commands when performing all other guidance AAs. Specific references to commands can be found throughout this AAR.

Component Testing Assurance Activities: The evaluator shall test the TOE's ability to correctly generate audit records by having the TOE generate audit records for the events listed in the table of audit events and administrative actions listed above. This should include all instances of an event: for instance, if there are several different I&A mechanisms for a system, the FIA_UIA_EXT.1 events must be generated for each mechanism. The evaluator shall test that audit records are generated for the establishment and termination of a channel for each of the cryptographic protocols contained in the ST. If HTTPS is implemented, the test demonstrating the establishment and termination of a TLS session can be combined with the test for an HTTPS session. When verifying the test results, the evaluator shall ensure the audit records generated during testing match the format specified in the guidance documentation, and that the fields in each audit record have the proper entries.

For distributed TOEs the evaluator shall perform tests on all TOE components according to the mapping of auditable events to TOE components in the Security Target. For all events involving more than one TOE component when an audit event is triggered, the evaluator has to check that the event has been audited on both sides (e.g. failure of building up a secure communication channel between the two components). This is not limited to error cases but includes also events about successful actions like successful build up/tear down of a secure communication channel between TOE components.

Note that the testing here can be accomplished in conjunction with the testing of the security mechanisms directly.

The evaluator created a list of the required audit events based on the Security Target requirements. The evaluator then collected audit events while running tests as part of other Assurance Activities described by the protection profile. For example, the required event for FPT_STM_EXT.1 is Changes to Time. The evaluator collected these audit records when modifying the clock using administrative commands. The evaluator then recorded these audit events in the proprietary Detailed Test Report (DTR). The security management events are handled in a similar manner. When the administrator was required to set a value for testing, the audit record associated with the administrator action was collected and recorded in the DTR.

2.1.2 Security Audit Data Generation (STFFW14e:FAU_GEN.1)

2.1.2.1 STFFW14e:FAU_GEN.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: No additional Evaluation Activities are specified.

No additional Evaluation Activities are specified.

Component Guidance Assurance Activities: In addition to the Evaluation Activities specified in the Supporting Document for the Base-PP, the evaluator shall check the guidance documentation to ensure that it describes the audit records specified in Table 2 of the PP-Module in addition to those required by the Base-PP. If the optional SFR FFW_RUL_EXT.2 is claimed by the TOE, the evaluator shall also check the guidance documentation to ensure that it describes the relevant audit record specified in Table 3 of the PP-Module.

See NDcPP22e:FAU_GEN.1 above which specifies where all audit records in the Guide have been identified including those specified in Table 2 and Table 3 of the PP-Module.

Component Testing Assurance Activities: In addition to the Evaluation Activities specified in the Supporting Document for the Base-PP, the evaluator shall perform tests to demonstrate that audit records are generated for the auditable events as specified in Table 2 of the PP-Module and, if the optional SFR FFW_RUL_EXT.2 is claimed by the TOE, Table 3.

The audit records associated with STFFW13:FFW_RUL_EXT.2 were collected during testing of that requirement. All required audits were found during that testing and the audits contained the necessary information.

2.1.3 Audit Data Generation (VPN Gateway) (VPNGW13:FAU_GEN.1/VPN)

2.1.3.1 VPNGW13:FAU_GEN.1.1/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.3.2 VPNGW13:FAU_GEN.1.2/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to verify that it describes the audit mechanisms that the TOE uses to generate audit records for VPN gateway behavior. If any audit mechanisms the TSF uses for this are not used to generate audit records for events defined by FAU_GEN.1 in the Base-PP, the evaluator shall ensure that any VPN gateway-specific audit mechanisms also meet the relevant functional claims from the Base-PP.

For example, FAU_STG_EXT.1 requires all audit records to be transmitted to the OE over a trusted channel.

This includes the audit records that are required by FAU_GEN.1/VPN. Therefore, if the TOE has an audit mechanism that is only used for VPN gateway functionality, the evaluator shall ensure that the VPN gateway related audit records meet this requirement, even if the mechanism used to generate these audit records does not apply to any of the auditable events defined in the Base-PP.

Section 6.1, Table 19 of [ST] provides the following:

FAU_GEN.1 row states that network interfaces have bandwidth limitations, and other traffic flow limitations that are configurable. When an interface has exceeded a limit for processing traffic, traffic will be dropped and audit messages can be generated. In order for events to be logged for information flow requests, the 'log' keyword must be in each line of an access control list.

Attempts to establish an IPsec tunnel or the failure of an established IPsec tunnel is logged as well as successfully established and terminated IPsec sessions with a peer. The connection to CA's or other entity (e.g. CDP) for the purpose of certificate verification or revocation check is logged. In addition, the TOE can be configured to capture the packets' contents during the session establishment.

FPF_RUL_EXT.1[VPN] row states that the TOE enforces information flow policies on network packets that are received by TOE interfaces and leave the TOE through other TOE interfaces. When network packets are received on a TOE interface, the TOE verifies whether the network traffic is allowed or not and performs one of the following actions, pass/not pass information, as well as optional logging.

Packets will be dropped unless a specific rule has been set up to allow the packet to pass (where the attributes of the packet match the attributes in the rule and the action associated with the rule is to pass traffic). Rules are enforced on a first match basis from the top down. As soon as a match is found the action associated with the rule is applied.

Component Guidance Assurance Activities: The evaluator shall examine the operational guidance to verify that it identifies all security-relevant auditable events claimed in the ST and includes sample records of each event type. If the TOE uses multiple audit mechanisms to generate different sets of records, the evaluator shall verify that the operational guidance identifies the audit records that are associated with each of the mechanisms such that the source of each audit record type is clear.

This activity has been addressed in conjunction with the guidance evaluation activities for VPNGW13:FPF_RUL_EXT.1 where the evaluator verified that the [AdminGuide] describes how to configure the TSF to result in applicable network traffic logging. In particular, see VPNGW13:FPF_RUL_EXT.1.4 which describes the 'access-list' command and how it is used to specify the actions of deny, permit or log for a rule.

Component Testing Assurance Activities: The evaluator shall test the audit functionality by performing actions that trigger each of the claimed audit events and verifying that the audit records are accurate and that their format is consistent with what is specified in the operational guidance. The evaluator may generate these audit events as a consequence of performing other tests that would cause these events to be generated.

Test 1: The evaluator attempted to flood the network and observed that an audit message was generated that correctly recorded the event.

Test 2: The evaluator used a VPN client to establish an IPsec session with the TOE and found that the event was logged.

2.1.4 User identity association (NDcPP22e:FAU_GEN.2)

2.1.4.1 NDcPP22e:FAU_GEN.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The TSS and Guidance Documentation requirements for FAU_GEN.2 are already covered by the TSS and Guidance Documentation requirements for FAU_GEN.1.

See NDcPP22e:FAU_GEN.1 where the activities for FAU_GEN.2 are already covered.

Component Guidance Assurance Activities: The TSS and Guidance Documentation requirements for FAU_GEN.2 are already covered by the TSS and Guidance Documentation requirements for FAU_GEN.1.

See NDcPP22e:FAU_GEN.1 where the activities for FAU_GEN.2 are already covered.

Component Testing Assurance Activities: This activity should be accomplished in conjunction with the testing of FAU_GEN.1.1.

For distributed TOEs the evaluator shall verify that where auditable events are instigated by another component, the component that records the event associates the event with the identity of the instigator. The evaluator shall perform at least one test on one component where another component instigates an auditable event. The evaluator shall verify that the event is recorded by the component as expected and the event is associated with the instigating component. It is assumed that an event instigated by another component can at least be generated for building up a secure channel between two TOE components. If for some reason (could be e.g. TSS or Guidance Documentation) the evaluator would come to the conclusion that the overall TOE does not generate any events instigated by other components, then this requirement shall be omitted.

See FAU_GEN.1.

2.1.5 Protected Audit Event Storage (NDcPP22e:FAU_STG_EXT.1)

2.1.5.1 NDcPP22e:FAU_STG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.5.2 NDcPP22e:FAU_STG_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.5.3 NDcPP22e:FAU_STG_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure it describes the means by which the audit data are transferred to the external audit server, and how the trusted channel is provided.

The evaluator shall examine the TSS to ensure it describes the amount of audit data that are stored locally; what happens when the local audit data store is full; and how these records are protected against unauthorized access.

The evaluator shall examine the TSS to ensure it describes whether the TOE is a standalone TOE that stores audit data locally or a distributed TOE that stores audit data locally on each TOE component or a distributed TOE that contains TOE components that cannot store audit data locally on themselves but need to transfer audit data to other TOE components that can store audit data locally. The evaluator shall examine the TSS to ensure that for

distributed TOEs it contains a list of TOE components that store audit data locally. The evaluator shall examine the TSS to ensure that for distributed TOEs that contain components which do not store audit data locally but transmit their generated audit data to other components it contains a mapping between the transmitting and storing TOE components.

The evaluator shall examine the TSS to ensure that it details the behavior of the TOE when the storage space for audit data is full. When the option 'overwrite previous audit record' is selected this description should include an outline of the rule for overwriting audit data. If 'other actions' are chosen such as sending the new audit data to an external IT entity, then the related behaviour of the TOE shall also be detailed in the TSS.

The evaluator shall examine the TSS to ensure that it details whether the transmission of audit information to an external IT entity can be done in real-time or periodically. In case the TOE does not perform transmission in real-time the evaluator needs to verify that the TSS provides details about what event stimulates the transmission to be made as well as the possible acceptable frequency for the transfer of audit data.

For distributed TOEs the evaluator shall examine the TSS to ensure it describes to which TOE components this SFR applies and how audit data transfer to the external audit server is implemented among the different TOE components (e.g. every TOE components does its own transfer or the data is sent to another TOE component for central transfer of all audit events to the external audit server).

For distributed TOEs the evaluator shall examine the TSS to ensure it describes which TOE components are storing audit information locally and which components are buffering audit information and forwarding the information to another TOE component for local storage. For every component the TSS shall describe the behaviour when local storage space or buffer space is exhausted.

Section 6.1, Table 19 (FAU_STG_EXT.1) in [ST] states that the TOE can be configured to export syslog records to an administrator-specified, external syslog server in real-time. The TOE can be configured to encrypt the communications with an external syslog server using IPsec or TLS.

The TOE will buffer syslog messages locally, but the local buffer will be cleared when the TOE is rebooted. The default size of the buffer is 4KB, and can be increased to 16KB. When the local buffer is full, the oldest message will be overwritten with new messages. Only authorized administrators can configure the local buffer size, reboot the TOE, and configure the external syslog server.

The TOE is not distributed.

Component Guidance Assurance Activities: The evaluator shall also examine the guidance documentation to ensure it describes how to establish the trusted channel to the audit server, as well as describe any requirements on the audit server (particular audit server protocol, version of the protocol required, etc.), as well as configuration of the TOE needed to communicate with the audit server.

The evaluator shall also examine the guidance documentation to determine that it describes the relationship between the local audit data and the audit data that are sent to the audit log server. For example, when an audit

event is generated, is it simultaneously sent to the external server and the local store, or is the local store used as a buffer and 'cleared' periodically by sending the data to the audit server.

The evaluator shall also ensure that the guidance documentation describes all possible configuration options for FAU_STG_EXT.1.3 and the resulting behaviour of the TOE for each possible configuration. The description of possible configuration options and resulting behaviour shall correspond to those described in the TSS.

Section “Secure Transmission of Audit Messages” in the [AdminGuide] instructs the user to ensure audit messages are transmitted securely from the ASA to the remote syslog server by configuring the connection to each syslog host to use IPsec or TLS to encrypt syslog messages as the messages leave the ASA.

Section “Securing Syslog with TLS” of the [AdminGuide] describes the use of the “logging host” command and an example for use of this command to enable RFC 6125 checks with an existing reference identity for a syslog server. The “Configuring TLS” section in the [AdminGuide] describes how to configure TLS.

Section “Securing Syslog with IPsec” in the [AdminGuide] indicates that to transmit the messages using IPsec, a crypto map to encrypt outbound syslog traffic with IPsec should be configured. The “Configuring IPsec” section in the [AdminGuide] describes how to configure IPsec.

Section “Configuring the Syslog Server” in the [AdminGuide] states that known compatible syslog servers include Kiwi Syslog Server release 9.2 or later; or syslog-ng release 2.0 or later.

Section “Logging and Log Messages” in the [AdminGuide] states that logging messages generated by the ASA can output to various destinations including the ‘console’ (which will include any connected CLI session, including SSH sessions), the local logging ‘buffer’ (which is a local circular log file with default size of 4KB that overwrites oldest messages when the log is full), and a logging ‘host’ (which is a remote syslog server). Logging to each of those destinations can be enabled or disabled independently, and each destination can be configured to receive a different set of log messages based on syslog severity level, or a ‘logging list” if one has been configured. Thus, the local logging buffer will not necessarily contain the same messages that are sent to a remote syslog server.

Section “Auditable Events Certified Under Common Criteria” in the [AdminGuide] indicates that when used with the external audit server configuration, generated audit events are simultaneously sent to the external server and the local logging buffer. The size of the local logging buffer can be configured using the ‘logging buffer-size’ command.

Component Testing Assurance Activities: Testing of the trusted channel mechanism for audit will be performed as specified in the associated assurance activities for the particular trusted channel mechanism. The evaluator shall perform the following additional test for this requirement:

a) Test 1: The evaluator shall establish a session between the TOE and the audit server according to the configuration guidance provided. The evaluator shall then examine the traffic that passes between the audit server and the TOE during several activities of the evaluator's choice designed to generate audit data to be transferred to the audit server. The evaluator shall observe that these data are not able to be viewed in the clear during this transfer, and that they are successfully received by the audit server. The evaluator shall record the particular

software (name, version) used on the audit server during testing. The evaluator shall verify that the TOE is capable of transferring audit data to an external audit server automatically without administrator intervention.

b) Test 2: The evaluator shall perform operations that generate audit data and verify that this data is stored locally. The evaluator shall perform operations that generate audit data until the local storage space is exceeded and verifies that the TOE complies with the behaviour defined in FAU_STG_EXT.1.3. Depending on the configuration this means that the evaluator has to check the content of the audit data when the audit data is just filled to the maximum and then verifies that

1) The audit data remains unchanged with every new auditable event that should be tracked but that the audit data is recorded again after the local storage for audit data is cleared (for the option 'drop new audit data' in FAU_STG_EXT.1.3).

2) The existing audit data is overwritten with every new auditable event that should be tracked according to the specified rule (for the option 'overwrite previous audit records' in FAU_STG_EXT.1.3)

3) The TOE behaves as specified (for the option 'other action' in FAU_STG_EXT.1.3).

c) Test 3: If the TOE complies with FAU_STG_EXT.2/LocSpace the evaluator shall verify that the numbers provided by the TOE according to the selection for FAU_STG_EXT.2/LocSpace are correct when performing the tests for FAU_STG_EXT.1.3.

d) Test 4: For distributed TOEs, Test 1 defined above should be applicable to all TOE components that forward audit data to an external audit server. For the local storage according to FAU_STG_EXT.1.2 and FAU_STG_EXT.1.3 the Test 2 specified above shall be applied to all TOE components that store audit data locally. For all TOE components that store audit data locally and comply with FAU_STG_EXT.2/LocSpace Test 3 specified above shall be applied. The evaluator shall verify that the transfer of audit data to an external audit server is implemented.

Test 1 - The evaluator configured the TOE to transfer audit data to a remote syslog server via TLS. The evaluator then began a packet capture of the traffic between the TOE and the syslog server, logged into the TOE, and performed 3 auditable actions. The evaluator issued an additional command to show the local logging buffer on the TOE and confirm these actions were audited. The evaluator then stopped the packet capture and observed that all audit data transferred to the remote server during the test was encrypted and could not be viewed in cleartext.

Test 2 - The evaluator configured the local logging buffer on the TOE to store a maximum of 50 audit logs, with any additional audits overwriting the oldest in storage. The evaluator logged into the TOE and printed the local logging buffer to observe that it was at maximum capacity. The evaluator then performed three auditable actions, the last of which printed the logging buffer once again. The evaluator observed that the previous three oldest audits had been overwritten and replaced in the logging buffer by the most recent audits.

2.2 Cryptographic support (FCS)

2.2.1 Cryptographic Key Generation (NDcPP22e:FCS_CKM.1)

2.2.1.1 NDcPP22e:FCS_CKM.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure that the TSS identifies the key sizes supported by the TOE. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme.

Section 6.1, Table 19 (FCS_CKM.1) in [ST] provides a table which identifies the usage for each scheme by mapping the scheme to SFRs and corresponding services. It states that in the TOE, cryptographic functions are used to establish TLS, HTTPS, SSH and IPsec sessions, for IPsec traffic and authentication keys, and for IKE authentication and encryption keys. Key generation for asymmetric keys on all models of the TOE implements ECDSA with NIST curve sizes P-256, P-384, and P-521 according to FIPS PUB 186-4, “Digital Signature Standard (DSS)”, Appendix B.4, RSA with key size 2048 and 3072 bits according to FIPS PUB 186-4, “Digital Signature Standard (DSS)”, Appendix B.3, FFC Schemes with key sizes 2048 and 3072 bits according to "Digital Signature Standard (DSS)", Appendix B.1 and using Diffie-Hellman group 14 that meet section 3 of RFC 3526.

The TOE provides cryptographic signature services using RSA and ECDSA with key sizes (modulus) of 2048 and 3072 bits, and 256, 384, and 521 bits, respectively. For RSA, the key size is configurable down to 1024, but only 2048 key size or higher is permitted in the evaluated configuration. The key generation is also tested as part of the signature generation and verification functions.

Component Guidance Assurance Activities: The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key generation scheme(s) and key size(s) for all cryptographic protocols defined in the Security Target.

The ”Managing Public Key Infrastructure (PKI) Keys” section of the [AdminGuide] describes how to generate keys using the ‘crypto key generate’ command to specify either an RSA key with key size 2048 or 3072 bits, or an ECDSA key type with curve size P-256, P-384 or P-521.

Component Testing Assurance Activities: Note: The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

Generation of long-term cryptographic keys (i.e. keys that are not ephemeral keys/session keys) might be performed automatically (e.g. during initial start-up). Testing of key generation must cover not only administrator invoked key generation but also automated key generation (if supported).

Key Generation for FIPS PUB 186-4 RSA Schemes

The evaluator shall verify the implementation of RSA Key Generation by the TOE using the Key Generation test. This test verifies the ability of the TSF to correctly produce values for the key components including the public

verification exponent e, the private prime factors p and q, the public modulus n and the calculation of the private signature exponent d.

Key Pair generation specifies 5 ways (or methods) to generate the primes p and q. These include:

a) Random Primes:

- Provable primes

- Probable primes

b) Primes with Conditions:

- Primes p1, p2, q1, q2, p and q shall all be provable primes

- Primes p1, p2, q1, and q2 shall be provable primes and p and q shall be probable primes

- Primes p1, p2, q1, q2, p and q shall all be probable primes

To test the key generation method for the Random Provable primes method and for all the Primes with Conditions methods, the evaluator must seed the TSF key generation routine with sufficient data to deterministically generate the RSA key pair. This includes the random seed(s), the public exponent of the RSA key, and the desired key length. For each key length supported, the evaluator shall have the TSF generate 25 key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation.

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test

For each supported NIST curve, i.e., P-256, P-384 and P-521, the evaluator shall require the implementation under test (IUT) to generate 10 private/public key pairs. The private key shall be generated using an approved random bit generator (RBG). To determine correctness, the evaluator shall submit the generated key pairs to the public key verification (PKV) function of a known good implementation.

FIPS 186-4 Public Key Verification (PKV) Test

For each supported NIST curve, i.e., P-256, P-384 and P-521, the evaluator shall generate 10 private/public key pairs using the key generation function of a known good implementation and modify five of the public key values so that they are incorrect, leaving five values unchanged (i.e., correct). The evaluator shall obtain in response a set of 10 PASS/FAIL values.

Key Generation for Finite-Field Cryptography (FFC)

The evaluator shall verify the implementation of the Parameters Generation and the Key Generation for FFC by the TOE using the Parameter Generation and Key Generation test. This test verifies the ability of the TSF to correctly

produce values for the field prime p, the cryptographic prime q (dividing p-1), the cryptographic group generator g, and the calculation of the private key x and public key y.

The Parameter generation specifies 2 ways (or methods) to generate the cryptographic prime q and the field prime p:

- Primes q and p shall both be provable primes

- Primes q and field prime p shall both be probable primes

and two ways to generate the cryptographic group generator g:

- Generator g constructed through a verifiable process

- Generator g constructed through an unverifiable process.

The Key generation specifies 2 ways to generate the private key x:

- len(q) bit output of RBG where 1 <=x <= q-1

- len(q) + 64 bit output of RBG, followed by a mod q-1 operation and a +1 operation, where 1<= x<=q-1.

The security strength of the RBG must be at least that of the security offered by the FFC parameter set.

To test the cryptographic and field prime generation method for the provable primes method and/or the group generator g for a verifiable process, the evaluator must seed the TSF parameter generation routine with sufficient data to deterministically generate the parameter set.

For each key length supported, the evaluator shall have the TSF generate 25 parameter sets and key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation. Verification must also confirm

- g != 0,1

- q divides p-1

- g^q mod p = 1

- g^x mod p = y

for each FFC parameter set and key pair.

FFC Schemes using 'safe-prime' groups

Testing for FFC Schemes using safe-prime groups is done as part of testing in CKM.2.1.

(TD0580 applied)

The TOE has been CAVP tested. Refer to the CAVP certificates identified in Section 1.1.2 CAVP Equivalence for test evidence associated with RSA, ECC and FFC key generation.

[ST] includes RSA, ECC, FFC and FFC Safe-primes.

Testing for FFC Schemes using 'safe-prime' groups is done as part of testing in CKM.2.1 where the TOE was tested against a known good implementation.

2.2.2 Cryptographic Key Generation (for IKE Peer Authentication) (VPNGW13:FCS_CKM.1/IKE)

2.2.2.1 VPNGW13:FCS_CKM.1.1/IKE

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check to ensure that the TSS describes how the key-pairs are generated. In order to show that the TSF implementation complies with FIPS PUB 186-4, the evaluator shall ensure that the TSS contains the following information:

- The TSS shall list all sections of Appendix B to which the TOE complies.

- For each applicable section listed in the TSS, for all statements that are not 'shall' (that is, 'shall not', 'should', and 'should not'), if the TOE implements such options it shall be described in the TSS. If the included functionality is indicated as 'shall not' or 'should not' in the standard, the TSS shall provide a rationale for why this will not adversely affect the security policy implemented by the TOE;

- For each applicable section of Appendix B, any omission of functionality related to 'shall' or 'should' statements shall be described;

Any TOE-specific extensions, processing that is not included in the Appendices, or alternative implementations allowed by the Appendices that may impact the security requirements the TOE is to enforce shall be described.

Section 6.1, Table 19 (FCS_CKM.1) of [ST] provides a table which identifies the usage for each scheme by mapping the scheme to SFRs and corresponding services. It states that in the TOE, cryptographic functions are used to establish TLS, HTTPS, IPsec and SSH sessions, for IPsec traffic and authentication keys, and for IKE authentication and encryption keys.

Key generation for asymmetric keys on all models of the TOE implements ECDSA with NIST curve sizes P-256, P-384, and P-521 according to FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.4, RSA with key size 2048 and 3072 bits according to FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3, FFC Schemes with key sizes 2048 and 3072 bits according to "Digital Signature Standard (DSS)", Appendix B.1 and using Diffie-

Hellman group 14 that meet section 3 of RFC 3526. This material also indicates that the TOE implements all “shall” and “should” statements and does not implement any “shall not” or “should not” statements from these appendix.

Component Guidance Assurance Activities: The evaluator shall check that the operational guidance describes how the key generation functionality is invoked, and describes the inputs and outputs associated with the process for each signature scheme supported. The evaluator shall also check that guidance is provided regarding the format and location of the output of the key generation process.

The “Managing Public Key Infrastructure (PKI) Keys” section of the [AdminGuide] describes how to generate keys using the ‘crypto key generate’ command to specify either an RSA key with key size 2048 bits, 3072 bits or an ECDSA key type with curve size P-256, P-384 or P-521. This section also describes how to display the key output set using the “show crypto key mypubkey [rsa | ecdsa]” command.

Component Testing Assurance Activities: For FFC Schemes using 'safe-prime' groups:

Testing for FFC Schemes using safe-prime groups is done as part of testing in FCS_CKM.2.

For all other selections:

The evaluator shall perform the corresponding tests for FCS_CKM.1 specified in the NDcPP SD, based on the selections chosen for this SFR. If IKE key generation is implemented by a different algorithm than the NDcPP key generation function, the evaluator shall ensure this testing is performed using the correct implementation.

The TOE has been CAVP tested. Refer to the CAVP certificates identified in section 1.1.2 CAVP Equivalence.

2.2.3 Cryptographic Key Establishment (NDcPP22e:FCS_CKM.2)

2.2.3.1 NDcPP22e:FCS_CKM.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure that the supported key establishment schemes correspond to the key generation schemes identified in FCS_CKM.1.1. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme. It is sufficient to provide the scheme, SFR, and service in the TSS.

The intent of this activity is to be able to identify the scheme being used by each service. This would mean, for example, one way to document scheme usage could be:

Scheme | SFR | Service

RSA | FCS_TLSS_EXT.1 | Administration

ECDH | FCS_SSHC_EXT.1 | Audit Server

ECDH | FCS_IPSEC_EXT.1 | Authentication Server

The information provided in the example above does not necessarily have to be included as a table but can be presented in other ways as long as the necessary data is available.

(TD0580 applied)

Section 6.1, Table 19 (FCS_CKM.2) in [ST] provides a table which identifies the usage for each scheme by mapping the scheme to SFRs and corresponding services. It shows that in the TOE, cryptographic functions are used to establish TLS, HTTPS, IPsec and SSH sessions, for IPsec traffic and authentication keys, and for IKE authentication and encryption keys.

Key establishment for asymmetric keys on all models of the TOE implements RSA-based, ECDSA-based and DH-based key establishment schemes as specified in Section 7.2 of RFC 3447, “Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1 and NIST SP 800-56A “Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography” respectively. In addition, the TOE also supports FFC schemes key establishment method that uses groups listed in RFC 3526.

The TOE provides cryptographic signature services using RSA and ECDSA with key sizes (modulus) of 2048 and 3072 bits, and 256, 384, and 521 bits, respectively. For RSA, the key size is configurable down to 1024, but only 2048 key size or higher is permitted in the evaluated configuration. The key generation is also tested as part of the signature generation and verification functions. The TOE supports key establishment including RSA-based, ECDSA-based and DH-based schemes.

Component Guidance Assurance Activities: The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key establishment scheme(s).

The ”Managing Public Key Infrastructure (PKI) Keys” section of the [AdminGuide] describes how to generate keys using the ‘crypto key generate’ command to specify either an RSA key with key size 2048 bits, 3072 bits or an ECDSA key type with curve size P-256, P-384 or P-521. The section “Create Trustpoint and Generate Certificate Signing Request (CSR)” of the [AdminGuide] describes the use of the selected key establishment schemes(s) in configuring a CA trustpoint.

In order to use a particular key establishment method, the TOE must already have generated a key appropriate for that establishment method, and thus the sections referenced not only generate keys, but define the establishment methods configured.

Component Testing Assurance Activities: Key Establishment Schemes

The evaluator shall verify the implementation of the key establishment schemes of the supported by the TOE using the applicable tests below.

SP800-56A Key Establishment Schemes

The evaluator shall verify a TOE's implementation of SP800-56A key agreement schemes using the following Function and Validity tests. These validation tests for each key agreement scheme verify that a TOE has implemented the components of the key agreement scheme according to the specifications in the Recommendation. These components include the calculation of the DLC primitives (the shared secret value Z) and the calculation of the derived keying material (DKM) via the Key Derivation Function (KDF). If key confirmation is supported, the evaluator shall also verify that the components of key confirmation have been implemented correctly, using the test procedures described below. This includes the parsing of the DKM, the generation of MACdata and the calculation of MACtag.

Function Test

The Function test verifies the ability of the TOE to implement the key agreement schemes correctly. To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each supported key agreement scheme-key agreement role combination, KDF type, and, if supported, key confirmation role- key confirmation type combination, the tester shall generate 10 sets of test vectors. The data set consists of one set of domain parameter values (FFC) or the NIST approved curve (ECC) per 10 sets of public keys. These keys are static, ephemeral or both depending on the scheme being tested.

The evaluator shall obtain the DKM, the corresponding TOE's public keys (static and/or ephemeral), the MAC tag(s), and any inputs used in the KDF, such as the Other Information field OI and TOE id fields.

If the TOE does not use a KDF defined in SP 800-56A, the evaluator shall obtain only the public keys and the hashed value of the shared secret.

The evaluator shall verify the correctness of the TSF's implementation of a given scheme by using a known good implementation to calculate the shared secret value, derive the keying material DKM, and compare hashes or MAC tags generated from these values.

If key confirmation is supported, the TSF shall perform the above for each implemented approved MAC algorithm.

Validity Test

The Validity test verifies the ability of the TOE to recognize another party's valid and invalid key agreement results with or without key confirmation. To conduct this test, the evaluator shall obtain a list of the supporting

cryptographic functions included in the SP800-56A key agreement implementation to determine which errors the TOE should be able to recognize. The evaluator generates a set of 24 (FFC) or 30 (ECC) test vectors consisting of data sets including domain parameter values or NIST approved curves, the evaluator's public keys, the TOE's public/private key pairs, MACTag, and any inputs used in the KDF, such as the other info and TOE id fields.

The evaluator shall inject an error in some of the test vectors to test that the TOE recognizes invalid key agreement results caused by the following fields being incorrect: the shared secret value Z, the DKM, the other information field OI, the data to be MACed, or the generated MACTag. If the TOE contains the full or partial (only ECC) public key validation, the evaluator will also individually inject errors in both parties' static public keys, both parties' ephemeral public keys and the TOE's static private key to assure the TOE detects errors in the public key validation function and/or the partial key validation function (in ECC only). At least two of the test vectors shall remain unmodified and therefore should result in valid key agreement results (they should pass).

The TOE shall use these modified test vectors to emulate the key agreement scheme using the corresponding parameters. The evaluator shall compare the TOE's results with the results using a known good implementation verifying that the TOE detects these errors.

RSA-based key establishment

The evaluator shall verify the correctness of the TSF's implementation of RSAES-PKCS1-v1_5 by using a known good implementation for each protocol selected in FTP_TRP.1/Admin, FTP_TRP.1/Join, FTP_ITC.1 and FPT_ITT.1 that uses RSAES-PKCS1-v1_5.

FFC Schemes using 'safe-prime' groups

The evaluator shall verify the correctness of the TSF's implementation of safe-prime groups by using a known good implementation for each protocol selected in FTP_TRP.1/Admin, FTP_TRP.1/Join, FTP_ITC.1 and FPT_ITT.1 that uses safe-prime groups. This test must be performed for each safe-prime group that each protocol uses.

(TD0580 applied)

The TOE has been CAVP tested. Refer to the CAVP certificates identified in section 1.1.2 CAVP Equivalence.

Gossamer tested against a 3rd party implementation to ensure the TOE is complaint with the RFC for Diffie-Hellman group 14 and 15. For TLS, that validation of a good implementation of Diffie-Hellman group 14 and Diffie-Hellman group15 was performed under FCS_TLSS_EXT.1.3 Test 2. During the use of this test, OpenSSL was leveraged to validate the TOE's implementation of a good implementation of these DH groups.

2.2.4 Cryptographic Key Destruction (NDcPP22e:FCS_CKM.4)

2.2.4.1 NDcPP22e:FCS_CKM.4.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator examines the TSS to ensure it lists all relevant keys (describing the origin and storage location of each), all relevant key destruction situations (e.g. factory reset or device wipe function, disconnection of trusted channels, key change as part of a secure channel protocol), and the destruction method used in each case. For the purpose of this Evaluation Activity the relevant keys are those keys that are relied upon to support any of the SFRs in the Security Target. The evaluator confirms that the description of keys and storage locations is consistent with the functions carried out by the TOE (e.g. that all keys for the TOE-specific secure channels and protocols, or that support FPT_APW.EXT.1 and FPT_SKP_EXT.1, are accounted for2). In particular, if a TOE claims not to store plaintext keys in non-volatile memory then the evaluator checks that this is consistent with the operation of the TOE.

The evaluator shall check to ensure the TSS identifies how the TOE destroys keys stored as plaintext in non-volatile memory, and that the description includes identification and description of the interfaces that the TOE uses to destroy keys (e.g., file system APIs, key store APIs).

Note that where selections involve 'destruction of reference' (for volatile memory) or 'invocation of an interface' (for non-volatile memory) then the relevant interface definition is examined by the evaluator to ensure that the interface supports the selection(s) and description in the TSS. In the case of non-volatile memory the evaluator includes in their examination the relevant interface description for each media type on which plaintext keys are stored. The presence of OS-level and storage device-level swap and cache files is not examined in the current version of the Evaluation Activity.

Where the TSS identifies keys that are stored in a non-plaintext form, the evaluator shall check that the TSS identifies the encryption method and the key-encrypting-key used, and that the key-encrypting-key is either itself stored in an encrypted form or that it is destroyed by a method included under FCS_CKM.4.

The evaluator shall check that the TSS identifies any configurations or circumstances that may not conform to the key destruction requirement (see further discussion in the Guidance Documentation section below). Note that reference may be made to the Guidance Documentation for description of the detail of such cases where destruction may be prevented or delayed.

Where the ST specifies the use of 'a value that does not contain any CSP' to overwrite keys, the evaluator examines the TSS to ensure that it describes how that pattern is obtained and used, and that this justifies the claim that the pattern does not contain any CSPs.

Section 6.1, Table 19 (FCS_CKM.4) of [ST] states that the TOE meets all requirements specified in FIPS 140-2 for destruction of keys and Critical Security Parameters (CSPs). As an example, Eexisting RSA and ECDSA keys will be zeroized when new RSA and ECDSA keys are generated, and zeroization of RSA and ECDSA keys can be triggered manually through use of the commands: ‘crypto key zeroized rsa’ and ‘crypto key zeroized ec’.

Section 7.2 of [ST] describes the key zeroization referenced by FCS_CKM.4 provided by the TOE for both volatile memory (DRAM) and non-volatile memory (NVRAM/flash). The table entitled "TOE Key Zeroization" within this section lists all of the Critical Security Parameters (CSPs) and indicates where each type of key material is stored

and when it is cleared. All CSPs are cleared using zeroization. For all CSPs in DRAM, the CSPs are zeroized via API calls or power cycle. For all CSPs in NVRAM, the CSPs are zeroized via commands that call APIs. The function used to zeroized the key buffers is 'crypto_key_zeroize()'. This function overwrites the key buffer with zeroes and verifies that the overwrite operation was successful. The evaluator confirmed that the table identifies the commands to zeroize all CSP that are stored in NVRAM, while also identifying the API used to clear all CSPs stored in DRAM.

Component Guidance Assurance Activities: A TOE may be subject to situations that could prevent or delay key destruction in some cases. The evaluator shall check that the guidance documentation identifies configurations or circumstances that may not strictly conform to the key destruction requirement, and that this description is consistent with the relevant parts of the TSS (and any other supporting information used). The evaluator shall check that the guidance documentation provides guidance on situations where key destruction may be delayed at the physical layer.

For example, when the TOE does not have full access to the physical memory, it is possible that the storage may be implementing wear-levelling and garbage collection. This may result in additional copies of the key that are logically inaccessible but persist physically. Where available, the TOE might then describe use of the TRIM command [Where TRIM is used then the TSS and/or guidance documentation is also expected to describe how the keys are stored such that they are not inaccessible to TRIM, (e.g. they would need not to be contained in a file less than 982 bytes which would be completely contained in the master file table)] and garbage collection to destroy these persistent copies upon their deletion (this would be explained in TSS and Operational Guidance).

[ST] states that the TOE meets all requirements specified in FIPS 140-2 for destruction of keys and Critical Security Parameters (CSPs). [ST] does not identify any circumstances or configurations where the key destruction does not conform to the requirement.

Section "Managing Public Key Infrastructure (PKI) Keys" in the [AdminGuide] indicates that in order to zeroize the key set, the following command should be used: "crypto key zeroize [rsa | ecdsa] [default | label <name> | noconfirm]". This section also notes that there are no configurations or circumstances that do not strictly conform to the key destruction requirement (FCS_CKM.4) as described in the Security Target.

Section "RSA Key Generation" in the [AdminGuide] also states that there are no configurations or circumstances that do not strictly conform to the key destruction requirement (FCS_CKM.4) as described in the Security Target.

Component Testing Assurance Activities: None Defined

2.2.5 Cryptographic Operation (AES Data Encryption/Decryption) (NDcPP22e:FCS_COP.1/DataEncryption)

2.2.5.1 NDcPP22e:FCS_COP.1.1/DataEncryption

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure it identifies the key size(s) and mode(s) supported by the TOE for data encryption/decryption.

Section 6.1, Table 19 (FCS_COP.1/DataEncryption) in [ST] states that the TOE's cryptographic functions are used to establish TLS, HTTPS, SSH and IPsec sessions, for IPsec traffic and authentication keys and for IKE authentication and encryption keys. The TOE supports AES-CBC and AES-GCM, each with 128 or 256-bit (as described in ISO 18033-3 for AES, ISO 10116 for CBC mode and ISO 19772 for GCM mode). The TOE uses a FIPS-validated implementation of AES with 128 and 256-bit keys.

Component Guidance Assurance Activities: The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected mode(s) and key size(s) defined in the Security Target supported by the TOE for data encryption/decryption.

The section entitled "Encryption Algorithms" in [AdminGuide] describes how an administrator can choose the encryption algorithm used for SSH encryption. This section indicates that when SSH version 2 is enabled, the ASA will support AES-CBC-128 and AES-CBC-256 which are permitted when FIPS mode is enabled.

The section entitled "Specify the TLS Ciphersuites (optional)" in [AdminGuide] explains that the “ssl cipher” command can be used to define the set of encryption algorithms that the ASA will allow to be negotiated for TLS sessions. This command defines the algorithms that will be allowed for both the ASA’s TLS client and the ASA’s TLS server. The evaluator noted that these ciphersuites include AES CBC and AES GCM with key sizes of 128-bit and 256-bit.

The sections entitled "IKEv2 Parameters for IKE Phase 1 (the IKE SA)” and "IKEv2 Parameters for IKE Phase 2 (the IPsec SA)" in [AdminGuide] identifies the commands used to specify the encryption used by the IKE SA and IPsec SA respectively. The evaluator noted that these instructions indicate IPsec and utilize AES CBC and AES GCM with keysizes of 128-bit and 256-bit.

Component Testing Assurance Activities: AES-CBC Known Answer Tests

There are four Known Answer Tests (KATs), described below. In all KATs, the plaintext, ciphertext, and IV values shall be 128-bit blocks. The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

KAT-1. To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 plaintext values and obtain the ciphertext value that results from AES-CBC encryption of the given plaintext using a key value of all zeros and an IV of all zeros. Five plaintext values shall be encrypted with a 128-bit all-zeros key, and the other five shall be encrypted with a 256-bit all-zeros key.

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using 10 ciphertext values as input and AES-CBC decryption.

KAT-2. To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 key values and obtain the ciphertext value that results from AES-CBC encryption of an all-zeros plaintext using the given key value and an IV of all zeros. Five of the keys shall be 128-bit keys, and the other five shall be 256-bit keys.

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using an all-zero ciphertext value as input and AES-CBC decryption.

KAT-3. To test the encrypt functionality of AES-CBC, the evaluator shall supply the two sets of key values described below and obtain the ciphertext value that results from AES encryption of an all-zeros plaintext using the given key value and an IV of all zeros. The first set of keys shall have 128 128-bit keys, and the second set shall have 256 256-bit keys. Key i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N].

To test the decrypt functionality of AES-CBC, the evaluator shall supply the two sets of keys and ciphertext value pairs described below and obtain the plaintext value that results from AES-CBC decryption of the given ciphertext using the given key and an IV of all zeros. The first set of key/ciphertext pairs shall have 128 128-bit key/ciphertext pairs, and the second set of key/ciphertext pairs shall have 256 256-bit key/ciphertext pairs. Key i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N]. The ciphertext value in each pair shall be the value that results in an all-zeros plaintext when decrypted with its corresponding key.

KAT-4. To test the encrypt functionality of AES-CBC, the evaluator shall supply the set of 128 plaintext values described below and obtain the two ciphertext values that result from AES-CBC encryption of the given plaintext using a 128-bit key value of all zeros with an IV of all zeros and using a 256-bit key value of all zeros with an IV of all zeros, respectively. Plaintext value i in each set shall have the leftmost i bits be ones and the rightmost 128-i bits be zeros, for i in [1,128].

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using ciphertext values of the same form as the plaintext in the encrypt test as input and AES-CBC decryption.

AES-CBC Multi-Block Message Test

The evaluator shall test the encrypt functionality by encrypting an i-block message where 1 < i <=10. The evaluator shall choose a key, an IV and plaintext message of length i blocks and encrypt the message, using the mode to be tested, with the chosen key and IV. The ciphertext shall be compared to the result of encrypting the same plaintext message with the same key and IV using a known good implementation.

The evaluator shall also test the decrypt functionality for each mode by decrypting an i-block message where 1 < i <=10. The evaluator shall choose a key, an IV and a ciphertext message of length i blocks and decrypt the message, using the mode to be tested, with the chosen key and IV. The plaintext shall be compared to the result of decrypting the same ciphertext message with the same key and IV using a known good implementation.

AES-CBC Monte Carlo Tests

The evaluator shall test the encrypt functionality using a set of 200 plaintext, IV, and key 3-tuples. 100 of these shall use 128 bit keys, and 100 shall use 256 bit keys. The plaintext and IV values shall be 128-bit blocks. For each 3-tuple, 1000 iterations shall be run as follows:

Input: PT, IV, Key

for i = 1 to 1000:

if i == 1:

CT[1] = AES-CBC-Encrypt(Key, IV, PT)

PT = IV

else:

CT[i] = AES-CBC-Encrypt(Key, PT)

PT = CT[i-1]

The ciphertext computed in the 1000th iteration (i.e., CT[1000]) is the result for that trial. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation.

The evaluator shall test the decrypt functionality using the same test as for encrypt, exchanging CT and PT and replacing AES-CBC-Encrypt with AES-CBC-Decrypt.

AES-GCM Test

The evaluator shall test the authenticated encrypt functionality of AES-GCM for each combination of the following input parameter lengths:

128 bit and 256 bit keys

a) Two plaintext lengths. One of the plaintext lengths shall be a non-zero integer multiple of 128 bits, if supported. The other plaintext length shall not be an integer multiple of 128 bits, if supported.

a) Three AAD lengths. One AAD length shall be 0, if supported. One AAD length shall be a non-zero integer multiple of 128 bits, if supported. One AAD length shall not be an integer multiple of 128 bits, if supported.

b) Two IV lengths. If 96 bit IV is supported, 96 bits shall be one of the two IV lengths tested.

The evaluator shall test the encrypt functionality using a set of 10 key, plaintext, AAD, and IV tuples for each combination of parameter lengths above and obtain the ciphertext value and tag that results from AES-GCM authenticated encrypt. Each supported tag length shall be tested at least once per set of 10. The IV value may be supplied by the evaluator or the implementation being tested, as long as it is known.

The evaluator shall test the decrypt functionality using a set of 10 key, ciphertext, tag, AAD, and IV 5-tuples for each combination of parameter lengths above and obtain a Pass/Fail result on authentication and the decrypted plaintext if Pass. The set shall include five tuples that Pass and five that Fail.

The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

AES-CTR Known Answer Tests

The Counter (CTR) mode is a confidentiality mode that features the application of the forward cipher to a set of input blocks, called counters, to produce a sequence of output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa. Due to the fact that Counter Mode does not specify the counter that is used, it is not possible to implement an automated test for this mode. The generation and management of the counter is tested through FCS_SSH*_EXT.1.4. If CBC and/or GCM are selected in FCS_COP.1/DataEncryption, the test activities for those modes sufficiently demonstrate the correctness of the AES algorithm. If CTR is the only selection in FCS_COP.1/DataEncryption, the AES-CBC Known Answer Test, AES-GCM Known Answer Test, or the following test shall be performed (all of these tests demonstrate the correctness of the AES algorithm):

There are four Known Answer Tests (KATs) described below to test a basic AES encryption operation (AES-ECB mode). For all KATs, the plaintext, IV, and ciphertext values shall be 128-bit blocks. The results from each test may either be obtained by the validator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

KAT-1 To test the encrypt functionality, the evaluator shall supply a set of 5 plaintext values for each selected keysize and obtain the ciphertext value that results from encryption of the given plaintext using a key value of all zeros.

KAT-2 To test the encrypt functionality, the evaluator shall supply a set of 5 key values for each selected keysize and obtain the ciphertext value that results from encryption of an all zeros plaintext using the given key value.

KAT-3 To test the encrypt functionality, the evaluator shall supply a set of key values for each selected keysize as described below and obtain the ciphertext values that result from AES encryption of an all zeros plaintext using the given key values. A set of 128 128-bit keys, a set of 192 192-bit keys, and/or a set of 256 256-bit keys. Key_i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1, N].

KAT-4 To test the encrypt functionality, the evaluator shall supply the set of 128 plaintext values described below and obtain the ciphertext values that result from encryption of the given plaintext using each selected keysize with a key value of all zeros (e.g. 256 ciphertext values will be generated if 128 bits and 256 bits are selected and 384 ciphertext values will be generated if all keysizes are selected). Plaintext value i in each set shall have the leftmost bits be ones and the rightmost 128-i bits be zeros, for i in [1, 128].

AES-CTR Multi-Block Message Test

The evaluator shall test the encrypt functionality by encrypting an i-block message where 1 less-than i less-than-or-equal to 10 (test shall be performed using AES-ECB mode). For each i the evaluator shall choose a key and plaintext message of length i blocks and encrypt the message, using the mode to be tested, with the chosen key. The ciphertext shall be compared to the result of encrypting the same plaintext message with the same key using a known good implementation. The evaluator shall perform this test using each selected keysize.

AES-CTR Monte-Carlo Test

The evaluator shall test the encrypt functionality using 100 plaintext/key pairs. The plaintext values shall be 128-bit blocks. For each pair, 1000 iterations shall be run as follows:

Input: PT, Key

for i = 1 to 1000:

CT[i] = AES-ECB-Encrypt(Key, PT) PT = CT[i]

The ciphertext computed in the 1000th iteration is the result for that trial. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation. The evaluator shall perform this test using each selected keysize.

There is no need to test the decryption engine.

The TOE has been CAVP tested. Refer to the CAVP certificates identified in section 1.1.2 CAVP Equivalence.

2.2.6 Cryptographic Operation (Hash Algorithm) (NDcPP22e:FCS_COP.1/Hash)

2.2.6.1 NDcPP22e:FCS_COP.1.1/Hash

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check that the association of the hash function with other TSF cryptographic functions (for example, the digital signature verification function) is documented in the TSS.

Section 6.1, Table 19, of [ST] provides the following:

FCS_COP.1/Hash states that the TOE provides cryptographic hashing services using SHA-1, SHA-256, SHA-384, and SHA-512, and keyed-hash message authentication using HMAC-SHA-1 (160-bit), HMAC-SHA-256 (256-bit), HMAC-SHA-384 (384-bit), and HMAC-SHA-512 (512-bit) with block size of 64 bytes (HMAC-SHA-1 and HMAC-SHA-256) and 128 bytes (HMAC-SHA-384 and HMAC-SHA-512).

FCS_TLSC_EXT.2 and FCS_TLSS_EXT.1 identify SHA1, SHA256 and SHA384 as part of the set of TLS ciphersuites supported by the TOE to implement HTTP over TLS (HTTPS) for remote administration using the ASDM, and TLS to support a secure syslog connection.

FCS_IPSEC_EXT.1 states the TOE implements IPsec using the ESP protocol as defined by RFC 4303, using the cryptographic algorithms AES-CBC-128, AES-CBC-256, AES-GCM-128 and AES-GCM-256 (both specified by RFCs 3602 and 4106) along with SHA-based HMAC algorithms (HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384 and HMAC-SHA-512). The SHA algorithms are configured by the administrator when defining the IPsec proposal and the IKEv2 policy.

This also include a claim that the negotiated pseudorandom function (prf) can be set by a 'prf' command so use sha1, sha256, sha384 or sha512.

FCS_SSHS_EXT.1 states that hashing algorithms hmac-sha1 and hmac-sha2-256 are supported by the TOE's SSHv2 implementation to ensure the integrity of the session.

FPT_TST_EXT.1 states that self-testing includes firmware integrity tests that verify the digital signature of the code image using RSA-2048 with SHA-512.

The claims in FCS_COP.1/HASH are consistent with claims regarding hashing functions in other requirements.

Component Guidance Assurance Activities: The evaluator checks the AGD documents to determine that any configuration that is required to configure the required hash sizes is present.

The section “Secure Communication” in the [AdminGuide] states that the Common Criteria certification evaluated the following cryptographic functionality, all of which must be configured as described in this guide. The following subsections then address "Enabling FIPS mode", "Disabling CiscoSSH", "Configuring SSH", "Configuring TLS" and "Configuring IPsec".

The sections entitled "Enabling FIPS mode" and "Disabling CiscoSSH" provide instructions to place the TOE into the proper configuration to force the use of FIPS supported encryption and an SSH implementation that can be configured to match Security Target claims.

The section entitled "Hashing Algorithms" explain that for SSH, when FIPS mode is enabled, only hmac-sha1 and hmac-sha2-256 are supported in the certified configuration. These algorithms match those claimed in the Security Target for SSHS operations.

The section entitled "Specify the TLS Version" states that ASA can be configured to restrict which TLS ciphersuites will be used by its TLS server (and client). In the certified configuration, the list of negotiated ciphersuites should be limited using the ssl cipher command and selecting one or more of these ciphersuites:

- DHE-RSA-AES128-SHA (TLS_DHE_RSA_WITH_AES_128_CBC_SHA)
- DHE-RSA-AES256-SHA (TLS_DHE_RSA_WITH_AES_256_CBC_SHA)
- DHE-RSA-AES128-SHA256 (TLS_DHE_RSA_WITH_AES_128_CBC_SHA256)
- DHE-RSA-AES256-SHA256 (TLS_DHE_RSA_WITH_AES_256_CBC_SHA256)

- ECDHE-ECDSA-AES128-GCM-SHA256 (TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256)
- ECDHE-ECDSA-AES256-GCM-SHA384 (TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384)

By specifying the ciphersuites the administrator is also configuring the Hashing algorithms used to support TLS. The ciphersuites shown match those claimed in the Security Target for TLS client and server operations.

The section entitled "IKEv2 Policies" describes commands that can be used to specify cryptographic algorithms used by IPsec. These include encryption algorithms, integrity algorithms and key exchange methods. The integrity algorithms shown match those claimed in the Security Target for IPsec.

Component Testing Assurance Activities: The TSF hashing functions can be implemented in one of two modes. The first mode is the byte-oriented mode. In this mode the TSF only hashes messages that are an integral number of bytes in length; i.e., the length (in bits) of the message to be hashed is divisible by 8. The second mode is the bit-oriented mode. In this mode the TSF hashes messages of arbitrary length. As there are different tests for each mode, an indication is given in the following sections for the bit-oriented vs. the byte-oriented testmacs.

The evaluator shall perform all of the following tests for each hash algorithm implemented by the TSF and used to satisfy the requirements of this PP.

Short Messages Test - Bit-oriented Mode

The evaluators devise an input set consisting of m+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m bits. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Short Messages Test - Byte-oriented Mode

The evaluators devise an input set consisting of m/8+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m/8 bytes, with each message being an integral number of bytes. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Selected Long Messages Test - Bit-oriented Mode

The evaluators devise an input set consisting of m messages, where m is the block length of the hash algorithm (e.g. 512 bits for SHA-256). The length of the ith message is m + 99*i, where 1 <= i <= m. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Selected Long Messages Test - Byte-oriented Mode

The evaluators devise an input set consisting of m/8 messages, where m is the block length of the hash algorithm (e.g. 512 bits for SHA-256). The length of the ith message is m + 8*99*i, where 1 <= i <= m/8. The message text

shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Pseudorandomly Generated Messages Test

This test is for byte-oriented implementations only. The evaluators randomly generate a seed that is n bits long, where n is the length of the message digest produced by the hash function to be tested. The evaluators then formulate a set of 100 messages and associated digests by following the algorithm provided in Figure 1 of [SHAVS]. The evaluators then ensure that the correct result is produced when the messages are provided to the TSF.

The TOE has been CAVP tested. Refer to the CAVP certificates identified in section 1.1.2 CAVP Equivalence.

2.2.7 Cryptographic Operation (Keyed Hash Algorithm) (NDcPP22e:FCS_COP.1/KeyedHash)

2.2.7.1 NDcPP22e:FCS_COP.1.1/KeyedHash

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure that it specifies the following values used by the HMAC function: key length, hash function used, block size, and output MAC length used.

Section 6.1, Table 19 (FCS_COP.1/KeyedHash) of [ST] states that the TOE provides cryptographic hashing services using SHA-1, SHA-256, SHA-384, and SHA-512, and keyed-hash message authentication using HMAC-SHA-1 (160-bit), HMAC-SHA-256 (256-bit), HMAC-SHA-384 (384-bit), and HMAC-SHA-512 (512-bit) with block size of 64 bytes (HMAC-SHA-1 and HMAC-SHA-256) and 128 bytes (HMAC-SHA-384 and HMAC-SHA-512).

Component Guidance Assurance Activities: The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the values used by the HMAC function: key length, hash function used, block size, and output MAC length used defined in the Security Target supported by the TOE for keyed hash function.

The section “Secure Communication” in the [AdminGuide] states that the Common Criteria certification evaluated the following cryptographic functionality, all of which must be configured as described in this guide. The following subsections then address "Enabling FIPS mode", "Disabling CiscoSSH", "Configuring SSH", "Configuring TLS" and "Configuring IPsec".

The following section references define the configuration of cryptographic hashing algorithms, which also dictates the TOE configuration for keyed-hashing algorithms used within the context of each protocol.

The sections entitled "Enabling FIPS mode" and "Disabling CiscoSSH" provide instructions to place the TOE into the proper configuration to force the use of FIPS supported encryption and an SSH implementation that can be configured to match Security Target claims.

The section entitled "Hashing Algorithms" explain that for SSH, when FIPS mode is enabled, only hmac-sha1 and hmac-sha2-256 are supported in the certified configuration. These algorithms match those claimed in the Security Target for SSHS operations.

The section entitled "Specify the TLS Version" states that ASA can be configured to restrict which TLS ciphersuites will be used by its TLS server (and client). In the certified configuration, the list of negotiated ciphersuites should be limited using the ssl cipher command and selecting one or more of these ciphersuites:

- DHE-RSA-AES128-SHA (TLS_DHE_RSA_WITH_AES_128_CBC_SHA)
- DHE-RSA-AES256-SHA (TLS_DHE_RSA_WITH_AES_256_CBC_SHA)
- DHE-RSA-AES128-SHA256 (TLS_DHE_RSA_WITH_AES_128_CBC_SHA256)
- DHE-RSA-AES256-SHA256 (TLS_DHE_RSA_WITH_AES_256_CBC_SHA256)
- ECDHE-ECDSA-AES128-GCM-SHA256 (TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256)
- ECDHE-ECDSA-AES256-GCM-SHA384 (TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384)

By specifying the ciphersuites the administrator is also configuring the keyed-hashing algorithms used to support TLS. The ciphersuites shown match those claimed in the Security Target for TLS client and server operations.

The section entitled "IKEv2 Policies" describes commands that can be used to specify cryptographic algorithms used by IPsec. These include encryption algorithms, integrity algorithms and key exchange methods. The integrity algorithms shown match those claimed in the Security Target for IPsec.

Component Testing Assurance Activities: For each of the supported parameter sets, the evaluator shall compose 15 sets of test data. Each set shall consist of a key and message data. The evaluator shall have the TSF generate HMAC tags for these sets of test data. The resulting MAC tags shall be compared to the result of generating HMAC tags with the same key and message data using a known good implementation.

The TOE has been CAVP tested. Refer to the CAVP certificates identified in section 1.1.2 CAVP Equivalence.

2.2.8 Cryptographic Operation (Signature Generation and Verification) (NDcPP22e:FCS_COP.1/SigGen)

2.2.8.1 NDcPP22e:FCS_COP.1.1/SigGen

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it specifies the cryptographic algorithm and key size supported by the TOE for signature services.

Section 6.1, Table 19 (FCS_COP.1/SigGen) of [ST] states that the TOE provides cryptographic signature services using RSA and ECDSA with key sizes (modulus) of 2048 and 3072 bits, and 256, 384, and 521 bits, respectively. For RSA, the key size is configurable down to 1024, but only 2048 key size or higher is permitted in the evaluated configuration. IKE/IPsec supports both ECDSA and RSA digital signature. SSH and trusted update only support RSA digital signature. The key generation is also tested as part of the signature generation and verification functions.

Component Guidance Assurance Activities: The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected cryptographic algorithm and key size defined in the Security Target supported by the TOE for signature services.

The TOE does not require explicit configuration to allow use of cryptographic signature services. Rather the TOE when put into fips mode, limits the cryptographic capabilities of the TOE to only those functions (including signature services) which are FIPS compliant as demonstrated by CAVP/ACVP testing.

The section entitled "Enabling FIPS mode" in [AdminGuide] provides instructions to place the TOE into the proper configuration to force the use of FIPS supported encryption. The use of FIPS mode is fully compatible with the Common Criteria evaluated configuration. Explicit configuration of cryptographic signature services is not necessary when the device is operating in FIPS mode; see [AdminGuide] section entitled “Evaluated Cryptography”.

Component Testing Assurance Activities: ECDSA Algorithm Tests

ECDSA FIPS 186-4 Signature Generation Test

For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate 10 1024-bit long messages and obtain for each message a public key and the resulting signature values R and S. To determine correctness, the evaluator shall use the signature verification function of a known good implementation.

ECDSA FIPS 186-4 Signature Verification Test

For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate a set of 10 1024-bit message, public key and signature tuples and modify one of the values (message, public key or signature) in five of the 10 tuples. The evaluator shall obtain in response a set of 10 PASS/FAIL values.

RSA Signature Algorithm Tests

Signature Generation Test

The evaluator generates or obtains 10 messages for each modulus size/SHA combination supported by the TOE. The TOE generates and returns the corresponding signatures.

The evaluator shall verify the correctness of the TOE's signature using a trusted reference implementation of the signature verification algorithm and the associated public keys to verify the signatures.

Signature Verification Test

For each modulus size/hash algorithm selected, the evaluator generates a modulus and three associated key pairs, (d, e). Each private key d is used to sign six pseudorandom messages each of 1024 bits using a trusted reference implementation of the signature generation algorithm. Some of the public keys, e, messages, or signatures are altered so that signature verification should fail. For both the set of original messages and the set of altered messages: the modulus, hash algorithm, public key e values, messages, and signatures are forwarded to the TOE, which then attempts to verify the signatures and returns the verification results.

The evaluator verifies that the TOE confirms correct signatures on the original messages and detects the errors introduced in the altered messages.

The TOE has been CAVP tested. Refer to the CAVP certificates identified in section 1.1.2 CAVP Equivalence.

2.2.9 HTTPS Protocol (NDcPP22e:FCS_HTTPS_EXT.1)

2.2.9.1 NDcPP22e:FCS_HTTPS_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.9.2 NDcPP22e:FCS_HTTPS_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.9.3 NDcPP22e:FCS_HTTPS_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS and determine that enough detail is provided to explain how the implementation complies with RFC 2818.

Section 6.1, Table 19 (FCS_HTTPS_EXT.1) of [ST] states that the TOE implements HTTP over TLS in support of remote administration using the ASDM. The TOE supports TLSv1.2 only. The description of this implementation complies with RFC 2818.

Component Guidance Assurance Activities: The evaluator shall examine the guidance documentation to verify it instructs the Administrator how to configure TOE for use as an HTTPS client or HTTPS server.

The section entitled "Enabling HTTPS Access" in [AdminGuide] describes the steps an administrator must perform in order to configure ASDM access (which uses HTTPS).

Component Testing Assurance Activities: This test is now performed as part of FIA_X509_EXT.1/Rev testing.

Tests are performed in conjunction with the TLS evaluation activities.

If the TOE is an HTTPS client or an HTTPS server utilizing X.509 client authentication, then the certificate validity shall be tested in accordance with testing performed for FIA_X509_EXT.1.

This test was performed as part of NDcPP22e:FTP_TRP.1/Admin where the packet capture was inspected and showed that the TOE uses HTTP over TLS to protect administrative communications.

2.2.10 IPsec Protocol - Per TD0800 (NDcPP22e:FCS_IPSEC_EXT.1)

2.2.10.1 NDcPP22e:FCS_IPSEC_EXT.1.1

TSS Assurance Activities: The evaluator shall examine the TSS and determine that it describes what takes place when a packet is processed by the TOE, e.g., the algorithm used to process the packet. The TSS describes how the SPD is implemented and the rules for processing both inbound and outbound packets in terms of the IPsec policy. The TSS describes the rules that are available and the resulting actions available after matching a rule. The TSS describes how those rules and actions form the SPD in terms of the BYPASS (e.g., no encryption), DISCARD (e.g., drop the packet), and PROTECT (e.g., encrypt the packet) actions defined in

RFC 4301.

As noted in section 4.4.1 of RFC 4301, the processing of entries in the SPD is non-trivial and the evaluator shall determine that the description in the TSS is sufficient to determine which rules will be applied given the rule structure implemented by the TOE. For example, if the TOE allows specification of ranges, conditional rules, etc., the evaluator shall determine that the description of rule processing (for both inbound and outbound packets) is sufficient to determine the action that will be applied, especially in the case where two different rules may apply. This description shall cover both the initial packets (that is, no SA is established on the interface or for that particular packet) as well as packets that are part of an established SA.

Section 6.1, Table 19 (FCS_IPSEC_EXT.1) of [ST] states that a crypto map (the Security Policy Definition) set can contain multiple entries, each with a different access list. The crypto map entries are searched in a top-down sequence - the TOE attempts to match the packet to the crypto access control list (ACL) specified in that entry. The crypto ACL can specify a single address or a range of address and the crypto map can be applied to an inbound interface or an outbound interface. When a packet matches a permit entry in a particular access list, the method of security in the corresponding crypto map of that interface is applied. If the crypto map entry is tagged as *IPsecisakmp*, IPsec is triggered. The traffic matching the permit crypto ACLs would then flow through the IPsec tunnel and be classified as PROTECTED. Traffic that does not match a permit crypto ACL or match a deny crypto

ACL in the crypto map, but is permitted by other ACLs on the interface is allowed to BYPASS the tunnel. Traffic that does not match a permit crypto ACL or match a deny crypto ACL in the crypto map, and is also blocked by other non-crypto ACLs on the interface would be DISCARDED.

Guidance Assurance Activities: The evaluator shall examine the guidance documentation to verify it instructs the Administrator how to construct entries into the SPD that specify a rule for processing a packet. The description includes all three cases â€“ a rule that ensures packets are encrypted/decrypted, dropped, and flow through the TOE without being encrypted. The evaluator shall determine that the description in the guidance documentation is consistent with the description in the TSS, and that the level of detail in the guidance documentation is sufficient to allow the administrator to set up the SPD in an unambiguous fashion. This includes a discussion of how ordering of rules impacts the processing of an IP packet.

The “Create an Access-List and Assigning to Crypto Map” Section of the [AdminGuide] describes how to construct entries in the SPD that specify rules for processing a packet. Instructions are provided to explain how to create an access-list to define the traffic to be encrypted/decrypted, and create a crypto map that references that access-list, and defines the rest of the IPsec SA parameters. If the traffic matches any of the permit ACLs, the traffic will be protected using IPsec. If the traffic matches any of the deny ACLs, the traffic will be bypassed and be subjected to the interface ACLs. If the traffic matches none of the interface ACLs, it will be dropped.

The “Access Lists” section of the [AdminGuide] states that the ‘access-list’ command operates on a first-match basis. Therefore, the last rule added to the access list is the last rule checked. Administrators must take note of this when entering the initial rules during the configuration, as it may impact the remainder of the rule parsing.

Testing Assurance Activities: The evaluator uses the guidance documentation to configure the TOE to carry out the following tests:

a) Test 1: The evaluator shall configure the SPD such that there is a rule for dropping a packet, encrypting a packet, and allowing a packet to flow in plaintext. The selectors used in the construction of the rule shall be different such that the evaluator can generate a packet and send packets to the gateway with the appropriate fields (fields that are used by the rule - e.g., the IP addresses, TCP/UDP ports) in the packet header. The evaluator performs both positive and negative test cases for each type of rule (e.g. a packet that matches the rule and another that does not match the rule). The evaluator observes via the audit trail, and packet captures that the TOE exhibited the expected behaviour: appropriate packets were dropped, allowed to flow without modification, encrypted by the IPsec implementation.

b) Test 2: The evaluator shall devise several tests that cover a variety of scenarios for packet processing. As with Test 1, the evaluator ensures both positive and negative test cases are constructed. These scenarios must exercise the range of possibilities for SPD entries and processing modes as outlined in the TSS and guidance documentation. Potential areas to cover include rules with overlapping ranges and conflicting entries, inbound and outbound packets, and packets that establish SAs as well as packets that belong to established SAs. The evaluator shall verify, via the audit trail and packet captures, for each scenario that the expected behavior is exhibited, and is consistent with both the TSS and the guidance documentation.

Test 1: The evaluator created rules for each type of SPD policy. The evaluator then verified the rules by establishing an IPsec tunnel and successfully establishing an administrator connection. It was observed that the Bypass rule for administrator traffic, the Permit rule for IPsec traffic and the Discard rule were all successfully enforced against the traffic going through the IPsec tunnel. The evaluator also confirmed that packets not matching a Permit, Bypass or Discard rule were ignored/dropped.

Test 2: The ordering of rules was tested and successfully demonstrated as part of VPNGW11:FPF_RUL_EXT.1.5, test 1. The evaluator found that the rules were all enforced as expected consistent with the TSS and Guidance documentation.

2.2.10.2 NDcPP22e:FCS_IPSEC_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The assurance activity for this element is performed in conjunction with the activities for FCS_IPSEC_EXT.1.1.

The evaluator uses the guidance documentation to configure the TOE to carry out the following tests:

The evaluator shall configure the SPD such that there is a rule for dropping a packet, encrypting a packet, and allowing a packet to flow in plaintext. The evaluator may use the SPD that was created for verification of FCS_IPSEC_EXT.1.1. The evaluator shall construct a network packet that matches the rule to allow the packet to flow in plaintext and send that packet. The evaluator should observe that the network packet is passed to the proper destination interface with no modification. The evaluator shall then modify a field in the packet header; such that it no longer matches the evaluator-created entries (there may be a 'TOE create'' final entry that discards packets that do not match any previous entries). The evaluator sends the packet and observes that the packet was dropped.

Refer to FCS_IPSEC_EXT.1.1-t1 for results that demonstrate packets not matching a rule are dropped by the TOE. Also refer to the FPF_RUL_EXT.1.5 (rule ordering) test results which exercise various combinations of access list rule being matched, and demonstrate that the proper application of access-list rules.

2.2.10.3 NDcPP22e:FCS_IPSEC_EXT.1.3

TSS Assurance Activities: The evaluator checks the TSS to ensure it states that the VPN can be established to operate in transport mode and/or tunnel mode (as identified in FCS_IPSEC_EXT.1.3).

Section 6.1, Table 19 (FCS_IPSEC_EXT.1) of [ST] states that the TOE supports both transport and tunnel modes. Transport mode is only supported for peer-to-peer IPsec connection while tunnel mode is supported for all VPN connections including remote access.

Guidance Assurance Activities: The evaluator shall confirm that the guidance documentation contains instructions on how to configure the connection in each mode selected.

In the “Configuring IPsec” section in the [AdminGuide], the subsection “Select Tunnel or Transport mode” describes how to select the mode for the IPsec channel using the crypto map map_name sequence_number set ikev2 mode [tunnel | transport | transport-require] command.

Testing Assurance Activities: The evaluator shall perform the following test(s) based on the selections chosen:

Test 1: If tunnel mode is selected, the evaluator uses the guidance documentation to configure the TOE to operate in tunnel mode and also configures a VPN peer to operate in tunnel mode. The evaluator configures the TOE and the VPN peer to use any of the allowable cryptographic algorithms, authentication methods, etc. to ensure an allowable SA can be negotiated. The evaluator shall then initiate a connection from the TOE to connect to the VPN peer. The evaluator observes (for example, in the audit trail and the captured packets) that a successful connection was established using the tunnel mode.

Test 2: If transport mode is selected, the evaluator uses the guidance documentation to configure the TOE to operate in transport mode and also configures a VPN peer to operate in transport mode. The evaluator configures the TOE and the VPN peer to use any of the allowed cryptographic algorithms, authentication methods, etc. to ensure an allowable SA can be negotiated. The evaluator then initiates a connection from the TOE to connect to the VPN peer. The evaluator observes (for example, in the audit trail and the captured packets) that a successful connection was established using the transport mode.

Test 1: For this test, the evaluator alternately configured a test peer to require only tunnel mode or only transport mode. In each case, the evaluator then attempted to connect the IPsec VPN between the test peer and the TOE and confirmed that the connection was successful in both tunnel and transport mode.

Test 2: This test was covered as part of test 1 where both tunnel mode and transport mode were confirmed to be supported by the TOE as evidenced by successful connections and the packet captures.

2.2.10.4 NDcPP22e:FCS_IPSEC_EXT.1.4

TSS Assurance Activities: The evaluator shall examine the TSS to verify that the algorithms are implemented. In addition, the evaluator ensures that the SHA-based HMAC algorithm conforms to the algorithms specified in FCS_COP.1(4)/KeyedHash Cryptographic Operations (for keyed-hash message authentication) and if the SHA-based HMAC function truncated output is utilized it must also be described.

Section 6.1, Table 19 (FCS_IPSEC_EXT.1) of [ST] states that the TOE implements IPsec using the ESP protocol as defined by RFC 4303, using the cryptographic algorithms AES-CBC-128, AES-CBC-256, AES-GCM-128 and AES-GCM-256 (both specified by RFCs 3602 and 4106) along with SHA-based HMAC algorithms (HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384 and HMAC-SHA-512). These selections are consistent with the algorithms specified in FCS_IPSEC_EXT.1.4 and FCS_COP.1/KeyedHash.

The TOE allows the administrator to define the IPsec proposal and IKEv2 policy for any IPsec connection to use specific encryption methods and authentication methods using the 'protocol esp integrity' and 'integrity' commands. When AES-GCM is used for encryption, the ESP integrity selection will be "null" because GCM mode provides integrity.

Guidance Assurance Activities: The evaluator checks the guidance documentation to ensure it provides instructions on how to configure the TOE to use the algorithms selected.

In section “Configuring IPsec” of the [AdminGuide], subsections “IKEv2 Parameters for IKE Phase 1 (the IKE SA)” and "IKEv2 Parameters for IKE Phase 2 (the IPsec SA)” describe how to configure the claimed algorithms. This includes AES-CBC-128, AES-CBC-256, AES-GCM-128, AES-GCM-256 as well as the required hash sizes SHA-1, SHA-256, SHA-384, and SHA-512. The IKE SA algorithms are specified using the 'crypto ikev2 policy' command and its 'encryption' and 'integrity' subcommands. The algorithms used in the IPsec SA are specified using the 'crypto IPsec ikev2 IPsec-proposal' command and its ‘protocol esp encryption’ and ‘protocol esp integrity’ commands.

Testing Assurance Activities: The evaluator shall configure the TOE as indicated in the guidance documentation configuring the TOE to use each of the supported algorithms, attempt to establish a connection using ESP, and verify that the attempt succeeds.

For this test, the evaluator alternately configured a test peer to accept each of the algorithms claimed and supported by the TOE. The evaluator then attempted to connect the IPsec VPN between the test peer and the TOE and confirmed that the connection was successful with each of the supported algorithms.

2.2.10.5 NDcPP22e:FCS_IPSEC_EXT.1.5

TSS Assurance Activities: The evaluator shall examine the TSS to verify that IKEv1 and/or IKEv2 are implemented.

For IKEv1 implementations, the evaluator shall examine the TSS to ensure that, in the description of the IPsec protocol, it states that aggressive mode is not used for IKEv1 Phase 1 exchanges, and that only main mode is used. It may be that this is a configurable option.

Section 6.1, Table 19 (FCS_IPSEC_EXT.1) of [ST] states that only IKEv2 is supported. The IKEv2 protocols implement Peer Authentication using the RSA and ECDSA algorithms with X.509v3 certificates, or pre-shared keys. IKEv2 separates negotiation into two phases: SA and Child SA. IKE SA creates the first tunnel, which protects later IKE negotiation messages. The key negotiated in IKE SA enables IKE peers to communicate securely in IKE Child SA. During Child SA IKE establishes the IPsec SA. IKE maintains a trusted channel, referred to as a Security Association (SA), between IPsec peers that is also used to manage IPsec connections.

NAT traversal is supported in IKEv2 by default.

The IKE SA exchanges use only main mode.

Administrators can require use of main mode by configuring the mode for each IPsec tunnel, using the ‘crypto-map’ command.

Guidance Assurance Activities: The evaluator shall check the guidance documentation to ensure it instructs the administrator how to configure the TOE to use IKEv1 and/or IKEv2 (as selected), and how to configure the TOE to perform NAT traversal for the following test (if selected).

If the IKEv1 Phase 1 mode requires configuration of the TOE prior to its operation, the evaluator shall check the guidance documentation to ensure that instructions for this configuration are contained within that guidance.

The TOE only supports IKEv2. The “Enable IKEv2” section of the [AdminGuide] provides the commands for enabling IKEv2 and for enabling NAT traversal so that ESP packets can pass through one or more NAT devices.

Testing Assurance Activities: Tests are performed in conjunction with the other IPsec evaluation activities.

a) Test 1: If IKEv1 is selected, the evaluator shall configure the TOE as indicated in the guidance documentation and attempt to establish a connection using an IKEv1 Phase 1 connection in aggressive mode. This attempt should fail. The evaluator should then show that main mode exchanges are supported.

b) Test 2: If NAT traversal is selected within the IKEv2 selection, the evaluator shall configure the TOE so that it will perform NAT traversal processing as described in the TSS and RFC 5996, section 2.23. The evaluator shall initiate an IPsec connection and determine that the NAT is successfully traversed.

Test 1: Not applicable. The TOE does not support IKEv1.

Test 2: The evaluator configured the TOE such that a VPN session from a test server traversed a NAT device. The evaluator initiated an IPsec connection and observed that the TOE correctly negotiated the NAT connection to establish a protected IPsec connection.

2.2.10.6 NDcPP22e:FCS_IPSEC_EXT.1.6

TSS Assurance Activities: The evaluator shall ensure the TSS identifies the algorithms used for encrypting the IKEv1 and/or IKEv2 payload, and that the algorithms chosen in the selection of the requirement are included in the TSS discussion.

Section 6.1, Table 19 (FCS_IPSEC_EXT.1) of [ST] indicates that for the IKEv2 protocols supported by the TOE the following command is used to specify the encryption algorithms used for SAs:

asa(config-ikev2-policy)#encryption [aes | aes-256 | aes-gcm | aes-gcm-256]

The ‘encryption’ command indicates that the aes-cbc-128, aes-cbc-256, aes-gcm-128 and aes-gcm-256 algorithms are supported as selected in FCS_IPSEC_EXT.1.6.

Guidance Assurance Activities: The evaluator ensures that the guidance documentation describes the configuration of all selected algorithms in the requirement.

In section “Configuring IPsec” of the [AdminGuide], subsections “IKEv2 Parameters for IKE Phase 1 (the IKE SA)” and "IKEv2 Parameters for IKE Phase 2 (the IPsec SA)” describe how to configure the claimed algorithms. This includes AES-CBC-128, AES-CBC-256, AES-GCM-128, AES-GCM-256 as well as the required hash sizes SHA-1, SHA-256, SHA-384, and SHA-512. The IKE SA algorithms are specified using the 'crypto ikev2 policy' command and its 'encryption' and 'integrity' subcommands. The algorithms used in the IPsec SA are specified using the 'crypto IPsec ikev2 IPsec-proposal' command and its ‘protocol esp encryption’ and ‘protocol esp integrity’ commands.

Testing Assurance Activities: The evaluator shall configure the TOE to use the ciphersuite under test to encrypt the IKEv1 and/or IKEv2 payload and establish a connection with a peer device, which is configured to only accept the payload encrypted using the indicated ciphersuite. The evaluator will confirm the algorithm was that used in the negotiation.

The evaluator configured IKEv2 profiles for AES-CBC-128, AES-CBC-256, AES-GCM-128 and AES-GCM-256 on the TOE. The evaluator then confirmed that the TOE could establish a session with each algorithm. For these tests the TOE and a test server were configured as peers and the evaluator configured each to use the same selected algorithms to establish the tunnel (the test was repeated for each algorithm).

2.2.10.7 NDcPP22e:FCS_IPSEC_EXT.1.7

TSS Assurance Activities: The evaluator shall ensure the TSS identifies the lifetime configuration method used for limiting the IKEv1 Phase 1 SA lifetime and/or the IKEv2 SA lifetime. The evaluator shall verify that the selection made here corresponds to the selection in FCS_IPSEC_EXT.1.5.

Section 6.1, Table 19 (FCS_IPSEC_EXT.1) of [ST] states that the IKE SA exchanges use only main mode and the IKE SA lifetimes are able to be limited to 24 hours for Phase 1 (SAs) and 8 hours for Phase 2 (Child SAs). IKEv2 SA can be limited by time only. IKEv2 Child SA can be limited by time or number of kilobytes (The valid range in kilobytes is 10 to 2,147,483,647 (10KB to 2TB). The time is in number of seconds. Administrators can require use of main mode by configuring the mode for each IPsec tunnel using the 'crypto map map-name seq-num set ikev2 phase1-mode main' command. This discussion regarding the IKEv2 SA lifetimes is consistent with the selection of IKEv2 in FCS_IPSEC_EXT.1.5.

Guidance Assurance Activities: The evaluator shall verify that the values for SA lifetimes can be configured and that the instructions for doing so are located in the guidance documentation. If time-based limits are supported, configuring the limit may lead to a rekey no later than the specified limit. For some implementations, it may be necessary, though, to configure the TOE with a lower time value to ensure a rekey is performed before the maximum SA lifetime of 24 hours is exceeded (e.g. configure a time value of 23h 45min to ensure the actual rekey is performed no later than 24h). The evaluator shall verify that the guidance documentation allows the Administrator to configure the Phase 1 SA value of 24 hours or provides sufficient instruction about the time value to configure to ensure the rekey is performed no later than the maximum SA lifetime of 24 hours. It is not permitted to configure a value of 24 hours if that leads to an actual rekey after more than 24hours. Currently there are no values mandated for the number of bytes, the evaluator just ensures that this can be configured if selected in the requirement. (TD0800 applied)

The “IKEv2 Parameters for IKE Phase 1 (the IKE SA)” section of the [AdminGuide] describes how to configure the IKE SA lifetime limits using the **lifetime seconds** *seconds* command. The “IKEv2 Parameters for IKE Phase 2 (the IPsec SA)” section describes how to set the lifetime limits for IKE and IPsec security associations using the **crypto IPsec security-association lifetime** {**seconds** *seconds* | **kilobytes** *kilobytes*} command. This section also notes that the actual rekey time may be 1 to 3 minutes longer than the configured value, so administrators should choose a value below 86100 to ensure rekeying occurs before 24 hours.

Testing Assurance Activities: When testing this functionality, the evaluator needs to ensure that both sides are configured appropriately. From the RFC 'A difference between IKEv1 and IKEv2 is that in IKEv1 SA lifetimes were negotiated. In IKEv2, each end of the SA is responsible for enforcing its own lifetime policy on the SA and rekeying the SA when necessary. If the two ends have different lifetime policies, the end with the shorter lifetime will end up always being the one to request the rekeying. If the two ends have the same lifetime policies, it is possible that both will initiate a rekeying at the same time (which will result in redundant SAs). To reduce the probability of this happening, the timing of rekeying requests SHOULD be jittered.'

Each of the following tests shall be performed for each version of IKE selected in the FCS_IPSEC_EXT.1.5 protocol selection:

a) Test 1: If 'number of bytes' is selected as the SA lifetime measure, the evaluator shall configure a maximum lifetime in terms of the number of bytes allowed following the guidance documentation. The evaluator shall configure a test peer with a byte lifetime that exceeds the lifetime of the TOE. The evaluator shall establish a SA between the TOE and the test peer, and determine that once the allowed number of bytes through this SA is exceeded, a new SA is negotiated. The evaluator shall verify that the TOE initiates a Phase 1 negotiation.

b) Test 2: If 'length of time' is selected as the SA lifetime measure, the evaluator shall configure a maximum lifetime no later than 24 hours for the Phase 1 SA following the guidance documentation. The evaluator shall configure a test peer with a Phase 1 SA lifetime that exceeds the lifetime Phase 1 SA on the TOE. The evaluator shall establish a SA between the TOE and the test peer, maintain the Phase 1 SA for 24 hours, and determine that a new Phase 1 SA is negotiated on or before 24 hours has elapsed. The evaluator shall verify that the TOE initiates a Phase 1 negotiation. (TD0800 applied)

Test 1: Not applicable. Number of bytes is not selected as an SA lifetime measure.

Test 2: The evaluator configured the TOE to have a 24 hour IKE and 8 hour ESP limits and the test peer was configured to have 25 hour IKE and 9 hour ESP limits. The evaluator then connected the IPsec VPN between the test peer and all instances of the TOE being tested and then waited for over 24 hours before terminating the test. The evaluator observed that the IKEv2 SA was renegotiated approximately 30 seconds before the 24 hour mark and that the IKEv2 Child SA renegotiation occurred approximately every 8 hours before the 8 hours elapsed.

2.2.10.8 NDcPP22e:FCS_IPSEC_EXT.1.8

TSS Assurance Activities: The evaluator shall ensure the TSS identifies the lifetime configuration method used for limiting the IKEv1 Phase 2 SA lifetime and/or the IKEv2 Child SA lifetime. The evaluator shall verify that the selection made here corresponds to the selection in FCS_IPSEC_EXT.1.5.

Section 6.1, Table 19 (FCS_IPSEC_EXT.1) of [ST] states that the IKE SA exchanges use only main mode and the IKE SA lifetimes are able to be limited to 24 hours for Phase 1 (SAs) and 8 hours for Phase 2 (Child SAs). IKEv2 SA can be limited by time only. IKEv2 Child SA can be limited by time or number of kilobytes (The valid range in kilobytes is 10 to 2,147,483,647 (10KB to 2TB). The time is in number of seconds. Administrators can require use of main mode by configuring the mode for each IPsec tunnel using the 'crypto map map-name seq-num set ikev2 phase1-mode

main' command. This discussion regarding the IKEv2 SA lifetimes is consistent with the selection of IKEv2 in FCS_IPSEC_EXT.1.5.

Guidance Assurance Activities: The evaluator shall verify that the values for SA lifetimes can be configured and that the instructions for doing so are located in the guidance documentation. If time-based limits are supported, configuring the limit may lead to a rekey no later than the specified limit. For some implementations, it may be necessary, though, to configure the TOE with a lower time value to ensure a rekey is performed before the maximum SA lifetime of 8 hours is exceeded (e.g. configure a time value of 7h 45min to ensure the actual rekey is performed no later than 8h). The evaluator shall verify that the guidance documentation allows the Administrator to configure the Phase 2 SA value of 8 hours or provides sufficient instruction about the time value to configure to ensure the rekey is performed no later than the maximum SA lifetime of 8 hours. It is not permitted to configure a value of 8 hours if that leads to an actual rekey after more than 8 hours. Currently there are no values mandated for the number of bytes, the evaluator just ensures that this can be configured if selected in the requirement. (TD0800 applied)

The “IKEv2 Parameters for IKE Phase 1 (the IKE SA)” section of the [AdminGuide] describes how to configure the IKE SA lifetime limits using the **lifetime seconds** *seconds* command. The “IKEv2 Parameters for IKE Phase 2 (the IPsec SA)” section describes how to set the lifetime limits for IKE and IPsec security associations using the **crypto IPsec security-association lifetime** {**seconds** *seconds* | **kilobytes** *kilobytes*} command.

Testing Assurance Activities: When testing this functionality, the evaluator needs to ensure that both sides are configured appropriately. From the RFC 'A difference between IKEv1 and IKEv2 is that in IKEv1 SA lifetimes were negotiated. In IKEv2, each end of the SA is responsible for enforcing its own lifetime policy on the SA and rekeying the SA when necessary. If the two ends have different lifetime policies, the end with the shorter lifetime will end up always being the one to request the rekeying. If the two ends have the same lifetime policies, it is possible that both will initiate a rekeying at the same time (which will result in redundant SAs). To reduce the probability of this happening, the timing of rekeying requests SHOULD be jittered.'

Each of the following tests shall be performed for each version of IKE selected in the FCS_IPSEC_EXT.1.5 protocol selection:

Test 1: If 'number of bytes' is selected as the SA lifetime measure, the evaluator shall configure a maximum lifetime in terms of the number of bytes allowed following the guidance documentation. The evaluator shall configure a test peer with a byte lifetime that exceeds the lifetime of the TOE. The evaluator shall establish a SA between the TOE and the test peer, and determine that once the allowed number of bytes through this SA is exceeded, a new SA is negotiated. The evaluator shall verify that the TOE initiates a Phase 2 negotiation.

Test 2: If 'length of time' is selected as the SA lifetime measure, the evaluator shall configure a maximum lifetime no later than 8 hours for the Phase 2 SA following the guidance documentation. The evaluator shall configure a test peer with a Phase 2 SA lifetime that exceeds the Phase 2 SA lifetime on the TOE. The evaluator shall establish a SA between the TOE and the test peer, maintain the Phase 1 SA for 8 hours, and determine that once a new Phase 2 SA is negotiated when or before 8 hours has elapsed. The evaluator shall verify that the TOE initiates a Phase 2 negotiation. (TD0800 applied)

Test 1: The evaluator configured a maximum lifetime in number of bytes on the TOE and then while the VPN test peer was configured without a limit. The evaluator then established a connection with the VPN peer. The IPsec SA timed out after the packet size was exceeded and the connection reset. A new Phase 2 SA negotiation was required.

Test 2: The testing of 'length of time' as the SA lifetime measure was performed during the NDcPP21:FCS_IPSEC_EXT.1.7-t2 test activity which showed ESP rekeying before 8 hours elapsed.

2.2.10.9 NDcPP22e:FCS_IPSEC_EXT.1.9

TSS Assurance Activities: The evaluator shall check to ensure that, for each DH group supported, the TSS describes the process for generating 'x'. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of 'x' meets the stipulations in the requirement.

Section 6.1, Table 19 (FCS_IPSEC_EXT.1) of [ST] states that the secret ‘x’ generated is 64 bytes long (or 512 bits), is the same across all the DH groups, and is generated with the DRBG specified in FCS_RBG_EXT.1. This is almost double the size of the highest comparable strength value which is 384 bits.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.10.10 NDcPP22e:FCS_IPSEC_EXT.1.10

TSS Assurance Activities: If the first selection is chosen, the evaluator shall check to ensure that, for each DH group supported, the TSS describes the process for generating each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of the nonces meet the stipulations in the requirement.

If the second selection is chosen, the evaluator shall check to ensure that, for each PRF hash supported, the TSS describes the process for generating each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of the nonces meet the stipulations in the requirement.

Section 6.1, Table 19 (FCS_IPSEC_EXT.1) of [ST] states that the secret ‘x’ (nonce) generated is 64 bytes long (or 512 bits), is the same across all the DH groups, and is generated with the DRBG specified in FCS_RBG_EXT.1. This is almost double the size of the highest comparable strength value which is 384 bits. The random number generator used to generate the nonces meets the requirements specified in FCS_RBG_EXT.1 for random bit generation. The TOE generates nonces used in IKEv2 exchanges, of at least 128 bits in size and at least half the output size of the negotiated pseudorandom function (PRF) hash.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: Each of the following tests shall be performed for each version of IKE selected in the FCS_IPSEC_EXT.1.5 protocol selection:

a) Test 1: If the first selection is chosen, the evaluator shall check to ensure that, for each DH group supported, the TSS describes the process for generating each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of the nonces meet the stipulations in the requirement.

b) Test 2: If the second selection is chosen, the evaluator shall check to ensure that, for each PRF hash supported, the TSS describes the process for generating each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of the nonces meet the stipulations in the requirement.

See TSS assurance activity above.

2.2.10.11 NDcPP22e:FCS_IPSEC_EXT.1.11

TSS Assurance Activities: The evaluator shall check to ensure that the DH groups specified in the requirement are listed as being supported in the TSS. If there is more than one DH group supported, the evaluator checks to ensure the TSS describes how a particular DH group is specified/negotiated with a peer.

Section 6.1, Table 19 (FCS_IPSEC_EXT.1) in [ST] states that the IKEv2 protocols supported by the TOE implement the following DH groups: 14 (2048-bit MODP), 19 (256-bit Random ECP), 20 (384-bit Random ECP), and use the RSA and ECDSA algorithms for Peer Authentication. The following command is used to specify the DH Group used for SAs.

asa(config-ikev2-policy) #group {14 | 19 | 20 }

Guidance Assurance Activities: The evaluator ensures that the guidance documentation describes the configuration of all algorithms selected in the requirement.

In section “Configuring IPsec” of the [AdminGuide], subsection “IKEv2 Parameters for IKE Phase 1 (the IKE SA)” describes how to configure the dh groups claimed in [ST] which are: dh groups 14, 19 and 20.

Testing Assurance Activities: For each supported DH group, the evaluator shall test to ensure that all supported IKE protocols can be successfully completed using that particular DH group.

Test 1: The evaluator made an IPsec connection to an IPsec peer using each of the claimed DH groups and confirmed that each resulted in successful connections.

2.2.10.12 NDcPP22e:FCS_IPSEC_EXT.1.12

TSS Assurance Activities: The evaluator shall check that the TSS describes the potential strengths (in terms of the number of bits in the symmetric key) of the algorithms that are allowed for the IKE and ESP exchanges. The TSS shall also describe the checks that are done when negotiating IKEv1 Phase 2 and/or IKEv2 CHILD_SA suites to ensure that the strength (in terms of the number of bits of key in the symmetric algorithm) of the negotiated algorithm is less than or equal to that of the IKE SA this is protecting the negotiation.

Section 6.1, Table 19 (FCS_IPSEC_EXT.1) of [ST] states that the TOE has a configuration option to deny tunnel if the phase 2 SA is weaker than the phase 1. The crypto strength check is enabled via the 'crypto IPsec ikev2 sa-strength-enforcement' command.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator simply follows the guidance to configure the TOE to perform the following tests.

a) Test 1: This test shall be performed for each version of IKE supported. The evaluator shall successfully negotiate an IPsec connection using each of the supported algorithms and hash functions identified in the requirements.

b) Test 2: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish a SA for ESP that selects an encryption algorithm with more strength than that being used for the IKE SA (i.e., symmetric algorithm with a key size larger than that being used for the IKE SA). Such attempts should fail.

c) Test 3: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an IKE SA using an algorithm that is not one of the supported algorithms and hash functions identified in the requirements. Such an attempt should fail.

d) Test 4: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish a SA for ESP (assumes the proper parameters where used to establish the IKE SA) that selects an encryption algorithm that is not identified in FCS_IPSEC_EXT.1.4. Such an attempt should fail.

Test 1: The evaluator alternately configured a test peer to accept each of the claimed IKE hash algorithm. In each case, the evaluator then attempted to connect the IPsec VPN between the test peer and the TOE and confirmed that the connections were successful

Test 2: The evaluator configured a test peer to use a 128-bit key size for IKE and 256-bit key size for ESP. The evaluator then attempted to connect the IPsec VPN between the test peer and the TOE and confirmed that the connection was rejected by the TOE.

Test 3: The evaluator attempted to establish a connection with an unsupported algorithm/hash combination. The connection attempt failed.

Test 4: The evaluator attempted to establish a connection with an unsupported ESP algorithm. The connection attempt failed.

2.2.10.13 NDcPP22e:FCS_IPSEC_EXT.1.13

TSS Assurance Activities: The evaluator ensures that the TSS identifies RSA and/or ECDSA as being used to perform peer authentication. The description must be consistent with the algorithms as specified in FCS_COP.1(2)/SigGen Cryptographic Operations (for cryptographic signature).

If pre-shared keys are chosen in the selection, the evaluator shall check to ensure that the TSS describes how pre-shared keys are established and used in authentication of IPsec connections. The description in the TSS shall also indicate how pre-shared key establishment is accomplished for TOEs that can generate a pre-shared key as well as TOEs that simply use a pre-shared key.

Section 6.1, Table 19 (FCS_IPSEC_EXT.1) of [ST] states that the TOE can be configured to authenticate IPsec connections using RSA and ECDSA signatures. This is consistent with the algorithms specified in FCS_COP.1/SigGen.

The IKEv2 protocols implement Peer Authentication using the RSA and ECDSA algorithm with X.509v3 certificates, or pre-shared keys. When using RSA and ECDSA signatures for authentication, the TOE and its peer must be configured to obtain certificates from the same certification authority (CA).

Pre-shared keys can be configured in the TOE for IPsec connection authentication. However, pre-shared keys are only supported when using IKEv2 for peer-to-peer VPNs. The text-based pre-shared keys can be 1-128 characters in length and are conditioned by a“prf” (pseudo-random function) configurable by the administrator. The bit-based pre-shared keys can be entered as HEX value as well. When using pre-shared keys for authentication, the IPsec endpoints must both be configured to use the same key.

Guidance Assurance Activities: The evaluator ensures the guidance documentation describes how to set up the TOE to use certificates with RSA and/or ECDSA signatures and public keys.

The evaluator shall check that the guidance documentation describes how pre-shared keys are to be generated and established. The description in the guidance documentation shall also indicate how pre-shared key establishment is accomplished for TOEs that can generate a pre-shared key as well as TOEs that simply use a pre-shared key.

The evaluator will ensure that the guidance documentation describes how to configure the TOE to connect to a trusted CA and ensure a valid certificate for that CA is loaded into the TOE and marked 'trusted'.

The ”Managing Public Key Infrastructure (PKI) Keys” section of the [AdminGuide] describes how to generate keys using the ‘crypto key generate’ command to specify either an RSA key with key size 2048 bits, 3072 bits, or an ECDSA key type with curve size P-256, P-384 or P-521.

The “Certificate Map Subject DN” section of the [AdminGuide] provides instructions for using the command ‘crypto ca certificate map’ to define certificate matching rules for IPsec tunnels.

The “Using X.509v3 Digital Certificates” section of the [AdminGuide] describes how the TOE supports X.509v3 certificates as defined by RFC 5280. The “Create Trustpoint and Generate Certificate Signing Request (CSR)” section describes how to generate an RSA or ECDSA key pair, create and configure a CA trustpoint, import the CA certificate for the configured trustpoint, generate the CSR and import the certificate to the TOE.

The “Using Pre-Shared Keys” section of the [AdminGuide] describes how preshared keys used for authentication of IPsec tunnels are entered by an administrator. The TOE does not generate pre-shared keys.

The "Post Quantum Preshared Key (PPK) Configuration" section of [AdminGuide] describes how Post Quantum Preshared Keys (PPKs) are included in the IKEv2 key material to make the exchange secure from quantum attacks.

Testing Assurance Activities: For efficiency sake, the testing is combined with the testing for FIA_X509_EXT.1, FIA_X509_EXT.2 (for IPsec connections), and FCS_IPSEC_EXT.1.1.

Test 1: Testing for NDcPP22e:FIA_X509_EXT.1/Rev was conducted using both RSA and ECDSA certificates. Testing using pre-shared keys was demonstrated as part of NDcPP22e:FCS_IPSEC_EXT.1.3. The combination of these tests demonstrate that the TOE is capable of performing peer authentication using ECDSA, RSA and pre-shared keys.

2.2.10.14 NDcPP22e:FCS_IPSEC_EXT.1.14

TSS Assurance Activities: The evaluator shall ensure that the TSS describes how the TOE compares the peer's presented identifier to the reference identifier. This description shall include which field(s) of the certificate are used as the presented identifier (DN, Common Name, or SAN). If the TOE simultaneously supports the same identifier type in the CN and SAN, the TSS shall describe how the TOE prioritizes the comparisons (e.g. the result of comparison if CN matches but SAN does not). If the location (e.g. CN or SAN) of non-DN identifier types must explicitly be configured as part of the reference identifier, the TSS shall state this. If the ST author assigned an additional identifier type, the TSS description shall also include a description of that type and the method by which that type is compared to the peer's presented certificate, including what field(s) are compared and which fields take precedence in the comparison.

Section 6.1, Table 19 (FCS_IPSEC_EXT.1) of [ST] explains that to define rules for matching the DN or FQDN of the IPsec peer certificate, use the 'crypto ca certificate map' command to create a certificate map with the mapping rules, then associate certificate map with the tunnel-group. It also indicates that a DN or FQDN is specified by defining a rule for "subject name" with attribute tag "san" to define a SAN (Subject Alternate name) mapping rule.

Guidance Assurance Activities: The evaluator shall ensure that the operational guidance describes all supported identifiers, explicitly states whether the TOE supports the SAN extension or not and includes detailed instructions on how to configure the reference identifier(s) used to check the identity of peer(s). If the identifier scheme implemented by the TOE does not guarantee unique identifiers, the evaluator shall ensure that the operational guidance provides a set of warnings and/or CA policy recommendations that would result in secure TOE use.

The “Certificate Map Subject DN” section of the [AdminGuide] describes how to indicate that a rule entry is applied to the subject DN or SAN:FQDN of the IPsec peer certificate by using the ‘subject-name’ command in ‘crypto ca certificate map’ configuration mode.

Testing Assurance Activities: In the context of the tests below, a valid certificate is a certificate that passes FIA_X509_EXT.1 validation checks but does not necessarily contain an authorized subject.

The evaluator shall perform the following tests:

Test 1: (conditional) For each CN/identifier type combination selected, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the field in the peer's presented

certificate and shall verify that the IKE authentication succeeds. If the TOE prioritizes CN checking over SAN (through explicit configuration of the field when specifying the reference identifier or prioritization rules), the evaluator shall also configure the SAN so it contains an incorrect identifier of the correct type (e.g. the reference identifier on the TOE is example.com, the CN=example.com, and the SAN:FQDN=otherdomain.com) and verify that IKE authentication succeeds.

Test 2: (conditional) For each SAN/identifier type combination selected, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the field in the peer's presented certificate and shall verify that the IKE authentication succeeds. If the TOE prioritizes SAN checking over CN (through explicit specification of the field when specifying the reference identifier or prioritization rules), the evaluator shall also configure the CN so it contains an incorrect identifier formatted to be the same type (e.g. the reference identifier on the TOE is DNS-ID; identify certificate has an identifier in SAN with correct DNS-ID, CN with incorrect DNS-ID (and not a different type of identifier)) and verify that IKE authentication succeeds.

Test 3: (conditional) For each CN/identifier type combination selected, the evaluator shall:

a) Create a valid certificate with the CN so it contains the valid identifier followed by ''. If the TOE prioritizes CN checking over SAN (through explicit specification of the field when specifying the reference identifier or prioritization rules) for the same identifier type, the evaluator shall configure the SAN so it matches the reference identifier.

b) Configure the peer's reference identifier on the TOE (per the administrative guidance) to match the CN without the '' and verify that IKE authentication fails.

Test 4: (conditional) For each SAN/identifier type combination selected, the evaluator shall:

a) Create a valid certificate with an incorrect identifier in the SAN. The evaluator shall configure a string representation of the correct identifier in the DN. If the TOE prioritizes CN checking over SAN (through explicit specification of the field when specifying the reference identifier or prioritization rules) for the same identifier type, the addition/modification shall be to any non-CN field of the DN. Otherwise, the addition/modification shall be to the CN.

b) Configure the peer's reference identifier on the TOE (per the administrative guidance) to match the correct identifier (expected in the SAN) and verify that IKE authentication fails.

Test 5: (conditional) If the TOE supports DN identifier types, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the subject DN in the peer's presented certificate and shall verify that the IKE authentication succeeds.

Test 6: (conditional) If the TOE supports DN identifier types, to demonstrate a bit-wise comparison of the DN, the evaluator shall create the following valid certificates and verify that the IKE authentication fails when each certificate is presented to the TOE:

a) Duplicate the CN field, so the otherwise authorized DN contains two identical CNs.

b) Append '' to a non-CN field of an otherwise authorized DN.

Test 1: Not applicable. The TOE does not claim support for CN/identifier type combinations as reference identifier.

Test 2: For this test, the evaluator configured a test peer to use an authentication certificate with the correct SAN: DNS address (FQDN) and then attempted to connect the IPsec VPN between the test peer and the TOE and confirmed that the connection was successful. The TOE does not utilize the CN in its checking, this is apparent because the CN used in this test does not match the DNS address (FQDN) of the peer, while the SAN: DNS value does match the DNS address of the peer. Therefore, the TOE accepts the certificate as valid authentication.

Test 3: Not applicable. The TOE does not claim support for CN/identifier type combinations as reference identifier.

Test 4: The evaluator configured a test peer to use a certificate that would present an incorrect SAN DNS (FQDN) reference identifier and a correct CN reference identifier. The evaluator attempted to connect the IPsec VPN between the test peer and the TOE and confirmed that the connection was rejected.

Test 5: The evaluator sent a peer certificate signed by a trusted CA with a DN that matches an expected DN. The evaluator observed that the TOE accepted the connection.

Test 6a: The evaluator sent a peer certificate signed by a trusted CA with a DN that matches an expected DN except that the DN contained a duplicate CN field. The evaluator observed that the TOE did not accept the connection.

Test 6b: The evaluator sent a peer certificate signed by a trusted CA with a DN that matches an expected DN except that the DN contained a NULL character within the DN. The evaluator observed that the TOE did not accept the connection.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.11 IPsec Protocol - per TD0824 (VPNGW13:FCS_IPSEC_EXT.1)

2.2.11.1 VPNGW13:FCS_IPSEC_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.2 VPNGW13:FCS_IPSEC_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.3 VPNGW13:FCS_IPSEC_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.4 VPNGW13:FCS_IPSEC_EXT.1.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.5 VPNGW13:FCS_IPSEC_EXT.1.5

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.6 VPNGW13:FCS_IPSEC_EXT.1.6

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.7 VPNGW13:FCS_IPSEC_EXT.1.7

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.8 VPNGW13:FCS_IPSEC_EXT.1.8

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.9 VPNGW13:FCS_IPSEC_EXT.1.9

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.10 VPNGW13:FCS_IPSEC_EXT.1.10

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.11 VPNGW13:FCS_IPSEC_EXT.1.11

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.12 VPNGW13:FCS_IPSEC_EXT.1.12

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.13 VPNGW13:FCS_IPSEC_EXT.1.13

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.14 VPNGW13:FCS_IPSEC_EXT.1.14

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: All existing activities regarding 'Pre-shared keys' apply to all selections including pre-shared keys. If any selection with 'Pre-shared keys' is included, the evaluator shall check to ensure that the TSS describes how the selection works in conjunction with the authentication of IPsec connections.

Section 6.1, Table 19 (FCS_COP.1/DataEncryption) of [ST] states that pre-shared keys can be configured in TOE for IPsec connection authentication. However, pre-shared keys are only supported when using IKEv2 for peer-to-peer VPNs.

Section 6.1, Table 19 (FIA_PSK_EXT.1) of [ST] states that the TOE supports use of IKEv2 pre-shared keys for authentication of IPsec tunnels

Component Guidance Assurance Activities: If any selection with 'Pre-shared Keys' is selected, the evaluator shall check that the operational guidance describes any configuration necessary to enable any selected authentication mechanisms.

The “Using Pre-Shared Keys” section of the [AdminGuide] describes how preshared keys used for authentication of IPsec tunnels are entered by an administrator. The TOE does not generate pre-shared keys.

The "Post Quantum Preshared Key (PPK) Configuration" section of [AdminGuide] describes how Post Quantum Preshared Keys (PPKs) are included in the IKEv2 key material to make the exchange secure from quantum attacks.

Component Testing Assurance Activities: None Defined

2.2.12 NTP Protocol (NDcPP22e:FCS_NTP_EXT.1)

2.2.12.1 NDcPP22e:FCS_NTP_EXT.1.1

TSS Assurance Activities: The evaluator shall examine the TSS to ensure it identifies the version of NTP supported, how it is implemented and what approach the TOE uses to ensure the timestamp it receives from an NTP timeserver (or NTP peer) is from an authenticated source and the integrity of the time has been maintained. The TOE must support at least one of the methods or may use multiple methods, as specified in the SFR element 1.2. The evaluator shall ensure that each method selected in the ST is described in the TSS, including the version of NTP supported in element 1.1, the message digest algorithms used to verify the authenticity of the timestamp and/or the protocols used to ensure integrity of the timestamp.

Section 6.1, Table 19 (FCS_NTP_EXT.1) of [ST] indicates that TOE implements an NTP client compliant with NTPv4 as defined by RFC 5905. The default NTPv4 is supported by the TOE and the NTP timestamp is not updated from broadcast or multicast addresses.

This section indicates that the TOE uses a SHA-1 message digest to authenticate the NTP time source.

This section also indicates that a maximum of 8 NTP time sources can be configured by an administrator.

Guidance Assurance Activities: The evaluator shall examine the guidance documentation to ensure it provides the Security Administrator instructions as how to configure the version of NTP supported, how to configure multiple NTP servers for the TOE's time source and how to configure the TOE to use the method(s) that are selected in the ST.

The section entitled "Configuring Network Time Protocol (NTP)" in [AdminGuide] indicates the Network Time Protocol (NTP) can be used to synchronize the clock of the ASA with a reliable time source. The ASA, by default, supports NTPv4.

This section also provides instructions to specify and authenticate NTP time sources, and states that a maximum of 8 NTP time sources can be configured by an administrator.

Testing Assurance Activities: The version of NTP selected in element 1.1 and specified in the ST shall be verified by observing establishment of a connection to an external NTP server known to be using the specified version(s) of NTP. This may be combined with tests of other aspects of FCS_NTP_EXT.1 as described below.

The evaluator configured the TOE to get NTP time updates from the evaluator's NTP Server and confirmed via packet capture that the TOE establishes a connection to the external NTP server using NTPv4 and the authentication algorithm as claimed in the ST.

2.2.12.2 NDcPP22e:FCS_NTP_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: For each of the secondary selections made in the ST, the evaluator shall examine the guidance document to ensure it instructs the Security Administrator how to configure the TOE to use the algorithms that support the authenticity of the timestamp and/or how to configure the TOE to use the protocols that ensure the integrity of the timestamp.

Assurance Activity Note:

Each primary selection in the SFR contains selections that specify a cryptographic algorithm or cryptographic protocol. For each of these secondary selections made in the ST, the evaluator shall examine the guidance documentation to ensure that the documentation instructs the administrator how to configure the TOE to use the chosen option(s).

The section entitled "Configuring Network Time Protocol (NTP)" in [AdminGuide] provides instructions to specify and authenticate NTP time sources.

Testing Assurance Activities: The cryptographic algorithms selected in element 1.2 and specified in the ST will have been specified in an FCS_COP SFR and tested in the accompanying Evaluation Activity for that SFR. Likewise, the cryptographic protocol selected in in element 1.2 and specified in the ST will have been specified in an FCS SFR and tested in the accompanying Evaluation Activity for that SFR.

[Conditional] If the message digest algorithm is claimed in element 1.2, the evaluator will change the message digest algorithm used by the NTP server in such a way that the new value does not match the configuration on the TOE and confirms that the TOE does not synchronize to this time source.

The evaluator shall use a packet sniffer to capture the network traffic between the TOE and the NTP server. The evaluator uses the captured network traffic, to verify the NTP version, to observe time change of the TOE and uses the TOE's audit log to determine that the TOE accepted the NTP server's timestamp update.

The captured traffic is also used to verify that the appropriate message digest algorithm was used to authenticate the time source and/or the appropriate protocol was used to ensure integrity of the timestamp that was transmitted in the NTP packets.

The evaluator configured the TOE to get NTP time updates from the evaluator's NTP Server. By changing the TOE and NTP server configuration the evaluator demonstrated that the TOE used the claimed message digest algorithm. The evaluator also configured the NTP Server to offer the wrong authentication data and observed that the TOE did not accept the time updates from the NTP Server.

2.2.12.3 NDcPP22e:FCS_NTP_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: The evaluator shall examine the guidance documentation to ensure it provides the Security Administrator instructions as how to configure the TOE to not accept broadcast and multicast NTP packets that would result in the timestamp being updated.

The section entitled "Configuring Network Time Protocol (NTP)" in [AdminGuide] states that the TOE, by default, does not accept time updates from broadcast and/or multicast addresses.

Testing Assurance Activities: The evaluator shall configure NTP server(s) to support periodic time updates to broadcast and multicast addresses. The evaluator shall confirm the TOE is configured to not accept broadcast and multicast NTP packets that would result in the timestamp being updated. The evaluator shall check that the time stamp is not updated after receipt of the broadcast and multicast packets.

The evaluator configured the TOE to get NTP time updates from the evaluator's NTP Server. The evaluator also configured the NTP server to send broadcast and multicast time updates such that they would be visible to the TOE. The evaluator observed that the TOE did not accept the time updates from the NTP Server.

2.2.12.4 NDcPP22e:FCS_NTP_EXT.1.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: Test 1: The evaluator shall confirm the TOE supports configuration of at least three (3) NTP time sources. The evaluator shall configure at least three NTP servers to support periodic time updates to the TOE. The evaluator shall confirm the TOE is configured to accept NTP packets that would result in the

timestamp being updated from each of the NTP servers. The evaluator shall check that the time stamp is updated after receipt of the NTP packets. The purpose of this test to verify that the TOE can be configured to synchronize with multiple NTP servers. It is up to the evaluator to determine that the multi- source update of the time information is appropriate and consistent with the behaviour prescribed by the RFC 1305 for NTPv3 and RFC 5905 for NTPv4.

Test 2: (The intent of this test is to ensure that the TOE would only accept NTP updates from configured NTP Servers).

The evaluator shall confirm that the TOE would not synchronize to other, not explicitly configured time sources by sending an otherwise valid but unsolicited NTP Server responses indicating different time from the TOE's current system time. This rogue time source needs to be configured in a way (e.g. degrade or disable valid and configured NTP servers) that could plausibly result in unsolicited updates becoming a preferred time source if they are not discarded by the TOE. The TOE is not mandated to respond in a detectable way or audit the occurrence of such unsolicited updates. The intent of this test is to ensure that the TOE would only accept NTP updates from configured NTP Servers. It is up to the evaluator to craft and transmit unsolicited updates in a way that would be consistent with the behaviour of a correctly-functioning NTP server.

(TD0528 applied)

Test 1: The evaluator configured the TOE to get NTP time updates from three (3) of the evaluator's NTP Servers. By changing the TOE and NTP server configuration the evaluator attempted to demonstrate that the TOE could use NTPv4 as claimed in the Security Target. Inspection of the network traffic between the TOE and the evaluator's NTP server revealed which versions of NTP the TOE supported and that time synchronization was successful.

Test 2: The evaluator referred to the results from FCS_NTP_EXT.1.3 where a properly configured NTP server sent time updates directly targeting the TOE. This NTP server is not configured as an authorized server in the TOE configuration. The TOE did not synchronize with the NTP server sending time updates.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.13 Random Bit Generation (NDcPP22e:FCS_RBG_EXT.1)

2.2.13.1 NDcPP22e:FCS_RBG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.13.2 NDcPP22e:FCS_RBG_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: Documentation shall be produced - and the evaluator shall perform the activities - in accordance with Appendix D of [NDcPP].

The evaluator shall examine the TSS to determine that it specifies the DRBG type, identifies the entropy source(s) seeding the DRBG, and state the assumed or calculated min-entropy supplied either separately by each source or the min-entropy contained in the combined seed value.

The Entropy description is provided in a separate (non-ST) Cisco proprietary document that has been delivered to NIAP for approval. Note that the entropy analysis has been accepted by NIAP/NSA.

Section 6.1, Table 19 (FCS_RBG_EXT.1) in [ST] states that the TOE uses a platform-based random bit generator that complies with ISO/IEC 18031:2011 using HMAC_DRBG(AES-256 Deterministic Random Bit Generation (DRBG) operating in FIPS mode. In addition, the DRBG is seeded by an entropy source that is at least 256-bit value derived from various highly sensitive and proprietary noise sources described in the proprietary Entropy Design document.

Component Guidance Assurance Activities: Documentation shall be produced - and the evaluator shall perform the activities - in accordance with Appendix D of [NDcPP].

The evaluator shall confirm that the guidance documentation contains appropriate instructions for configuring the RNG functionality.

The Entropy description is provided in a separate (non-ST) Cisco proprietary document that has been delivered to NIAP for approval. Note that the entropy analysis has been accepted by NIAP/NSA. Explicit configuration of RNG functionality is not necessary when the device is operating in FIPS mode; see [AdminGuide] section entitled “Evaluated Cryptography”.

Component Testing Assurance Activities: The evaluator shall perform 15 trials for the RNG implementation. If the RNG is configurable, the evaluator shall perform 15 trials for each configuration.

If the RNG has prediction resistance enabled, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) generate a second block of random bits (4) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 - 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The next two are additional input and entropy input for the first call to generate. The final two are additional input and entropy input for the second call to generate. These values are randomly generated. 'generate one block of random bits' means to generate random bits with number of returned bits equal to the Output Block Length (as defined in NIST SP800-90A).

If the RNG does not have prediction resistance, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) reseed, (4) generate a second block of random bits (5) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 - 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The fifth value is additional input to the first call to generate. The sixth and seventh are additional input and entropy input to the call to reseed. The final value is additional input to the second generate call.

The following paragraphs contain more information on some of the input values to be generated/selected by the evaluator.

Entropy input: the length of the entropy input value must equal the seed length.

Nonce: If a nonce is supported (CTR_DRBG with no Derivation Function does not use a nonce), the nonce bit length is one-half the seed length.

Personalization string: The length of the personalization string must be <= seed length. If the implementation only supports one personalization string length, then the same length can be used for both values. If more than one string length is support, the evaluator shall use personalization strings of two different lengths. If the implementation does not use a personalization string, no value needs to be supplied.

Additional input: the additional input bit lengths have the same defaults and restrictions as the personalization string lengths.

The TOE has been CAVP tested. Refer to the CAVP certificates identified in section 1.1.2 CAVP Equivalence.

2.2.14 SSH Server Protocol - per TD0631 (NDcPP22e:FCS_SSHS_EXT.1)

2.2.14.1 NDcPP22e:FCS_SSHS_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.14.2 NDcPP22e:FCS_SSHS_EXT.1.2

TSS Assurance Activities: The evaluator shall check to ensure that the TSS contains a list of supported public key algorithms that are accepted for client authentication and that this list is consistent with signature verification algorithms selected in FCS_COP.1/SigGen (e.g., accepting EC keys requires corresponding Elliptic Curve Digital Signature algorithm claims).

The evaluator shall confirm that the TSS includes the description of how the TOE establishes a user identity when an SSH client presents a public key or X.509v3 certificate. For example, the TOE could verify that the SSH client's presented public key matches one that is stored within the SSH server's authorized_keys file.

If password-based authentication method has been selected in the FCS_SSHS_EXT.1.2, then the evaluator shall confirm its role in the authentication process is described in the TSS. (TD0631 applied)

Section 6.1, Table 19 (FCS_SSHS_EXT.1) in [ST] explains that the TOE's implementation of SSHv2 supports the ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, or ecdsa-sha2-nistp521 algorithms for public key-based authentication for administrative users. This is consistent with the signature verification algorithm selected in FCS_COP.1/SigGen. The TOE ensures and verifies that the SSH client's presented public key matches one that is stored within the TOE's SSH server's authorized keys file.

This section explains that the TOE's SSH server's authorized keys file is stores the public keys that are mapped to admin accounts. The TOE's SSHv2 implementation also supports password-based authentication for administrative users.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: Test objective: The purpose of these tests is to verify server supports each claimed client authentication method.

Test 1: For each supported client public-key authentication algorithm, the evaluator shall configure a remote client to present a public key corresponding to that authentication method (e.g., 2048-bit RSA key when using ssh-rsa public key). The evaluator shall establish sufficient separate SSH connections with an appropriately configured remote non-TOE SSH client to demonstrate the use of all applicable public key algorithms. It is sufficient to observe the successful completion of the SSH Authentication Protocol to satisfy the intent of this test.

Test 2: The evaluator shall choose one client public key authentication algorithm supported by the TOE. The evaluator shall generate a new client key pair for that supported algorithm without configuring the TOE to recognize the associated public key for authentication. The evaluator shall use an SSH client to attempt to connect to the TOE with the new key pair and demonstrate that authentication fails.

Test 3: [Conditional] If password-based authentication method has been selected in the FCS_SSHS_EXT.1.2, the evaluator shall configure the TOE to accept password-based authentication and demonstrate that user authentication succeeds when the correct password is provided by the connecting SSH client.

Test 4: [Conditional] If password-based authentication method has been selected in the FCS_SSHS_EXT.1.2, the evaluator shall configure the TOE to accept password-based authentication and demonstrate that user authentication fails when the incorrect password is provided by the connecting SSH client.

(TD0631 applied)

Test 1: The evaluator configured a user to be able to login using the SSH interface with public-key based authentication, and observed the user login was successful.

Test 2: The evaluator attempted to login using the SSH interface with public-key authentication without configuring a public key for that user and observed that the login attempt was not successful.

Test 3: The evaluator configured the TOE for password authentication on the SSH interface. The evaluator logged in using an SSH client and the correct password. The login was successful.

Test 4: The evaluator attempted an SSH connection using an invalid password. The evaluator was not able to login.

2.2.14.3 NDcPP22e:FCS_SSHS_EXT.1.3

TSS Assurance Activities: The evaluator shall check that the TSS describes how 'large packets' in terms of RFC 4253 are detected and handled.

Section 6.1, Table 19 (FCS_SSHS_EXT.1) of [ST] states that SSH connections will be dropped if the TOE receives a packet larger than 32,750 bytes.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall demonstrate that if the TOE receives a packet larger than that specified in this component, that packet is dropped.

The evaluator created and sent a packet to the TOE that was larger than the maximum packet size of 32750 bytes. The TOE rejected the packet and the connection was closed.

2.2.14.4 NDcPP22e:FCS_SSHS_EXT.1.4

TSS Assurance Activities: The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that optional characteristics are specified, and the encryption algorithms supported are specified as well. The evaluator shall check the TSS to ensure that the encryption algorithms specified are identical to those listed for this component.

Section 6.1, Table 19 (FCS_SSHS_EXT.1) of [ST] states that the TOE's implementation of SSHv2 supports the encryption algorithms, AES-CBC-128 and AES-CBC-256 to ensure confidentiality of the session. This is consistent with the algorithms specified in FCS_SSHS_EXT.1.4.

Guidance Assurance Activities: The evaluator shall also check the guidance documentation to ensure that it contains instructions on configuring the TOE so that SSH conforms to the description in the TSS (for instance, the set of algorithms advertised by the TOE may have to be restricted to meet the requirements).

Section "Secure Communications" in the [AdminGuide] indicates that the TOE must be running in FIPS mode, that SSHv2 must be used instead of SSHv1, and that CiscoSSH must be disabled. The following subsections provide detailed instructions to accomplish these steps. Section "Enabling FIPS mode" provides instructions to use the "fips enable" command. The "Enable SSHv2 and Disable SSHv1" section in the [AdminGuide] provides instructions for enabling SSHv2.

The section "Enable SSHv2 and Disable SSHv1" states that when FIPS mode is enabled, the number of supported algorithms will be reduced to only the FIPS and CC Approved algorithms (AES128-CBC, AES256-CBC, HMAC-SHA1

and HMAC-SHA2-256). The section “Encryption Algorithms” in the [AdminGuide] provides instructions to configure specific algorithms should to ensure only desired algorithms are used (e.g., an administrator prefers to disable 128-bit algorithms).

Testing Assurance Activities: The evaluator must ensure that only claimed ciphers and cryptographic primitives are used to establish an SSH connection. To verify this, the evaluator shall start session establishment for an SSH connection from a remote client (referred to as 'remote endpoint' below). The evaluator shall capture the traffic exchanged between the TOE and the remote endpoint during protocol negotiation (e.g. using a packet capture tool or information provided by the endpoint, respectively). The evaluator shall verify from the captured traffic that the TOE offers all the ciphers defined in the TSS for the TOE for SSH sessions, but no additional ones compared to the definition in the TSS. The evaluator shall perform one successful negotiation of an SSH session to verify that the TOE behaves as expected. It is sufficient to observe the successful negotiation of the session to satisfy the intent of the test. If the evaluator detects that not all ciphers defined in the TSS for SSH are supported by the TOE and/or the TOE supports one or more additional ciphers not defined in the TSS for SSH, the test shall be regarded as failed.

The evaluator attempted to establish an SSH connection with each of the following SSH algorithms to encrypt the session: AES128-CBC and AES256-CBC. The evaluator captured packets associated with each of the connection attempts and observed that using these algorithms the evaluator was able to successfully connect to the TOE.

2.2.14.5 NDcPP22e:FCS_SSHS_EXT.1.5

TSS Assurance Activities: The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the SSH server's host public key algorithms supported are specified and that they are identical to those listed for this component. (TD0631 applied)

Section 6.1, Table 19 (FCS_SSHS_EXT.1) of [ST] states that the TOE’s implementation of SSHv2 supports the public key algorithms ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, ecdsa-sha2-nistp521. This is consistent with algorithm specified in FCS_SSHS_EXT.1.5.

Guidance Assurance Activities: The evaluator shall also check the guidance documentation to ensure that it contains instructions on configuring the TOE so that SSH conforms to the description in the TSS (for instance, the set of algorithms advertised by the TOE may have to be restricted to meet the requirements).

Section “ASA Installation” in the [AdminGuide] indicates that the TOE must be running in FIPS mode which is configured using the “fips enable” command. The “Enable SSHv2 and Disable SSHv1” section in the [AdminGuide] describes how to restrict the TOE to use SSHv2. When the TOE is configured with SSHv2 and FIPS mode, the security algorithms and ciphers available on the TOE are restricted.

The “ECDSA Key Generation” section of the [AdminGuide] describes how to generate an ECDSA key pairs using the “crypto key generate” command and instructs the user to use elliptic-curves 256, 384, or 521.

Testing Assurance Activities: Test objective: This test case is meant to validate that the TOE server will support host public keys of the claimed algorithm types.

Test 1: The evaluator shall configure (only if required by the TOE) the TOE to use each of the claimed host public key algorithms. The evaluator will then use an SSH client to confirm that the client can authenticate the TOE server public key using the claimed algorithm. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

Has effectively been moved to FCS_SSHS_EXT.1.2.

Test objective: This negative test case is meant to validate that the TOE server does not support host public key algorithms that are not claimed.

Test 2: The evaluator shall configure a non-TOE SSH client to only allow it to authenticate an SSH server host public key algorithm that is not included in the ST selection. The evaluator shall attempt to establish an SSH connection from the non-TOE SSH client to the TOE SSH server and observe that the connection is rejected.

(TD0631 applied)

Test 1: The evaluator attempted to establish an SSH connection with each of the following SSH public key algorithms: ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, and ecdsa-sha2-nistp521. The evaluator captured packets associated with each of the connection attempts. The evaluator observed that each algorithm was able to successfully connect to the TOE.

Test 2: The evaluator generated a new RSA key pair on a client and did not configure the TOE to recognize that key pair. The subsequent connection attempt failed.

Test 3: The evaluator attempted to establish an SSH connection using ssh-dsa. The evaluator captured packets and was able to determine the connection attempt failed as expected.

2.2.14.6 NDcPP22e:FCS_SSHS_EXT.1.6

TSS Assurance Activities: The evaluator shall check the TSS to ensure that it lists the supported data integrity algorithms, and that that list corresponds to the list in this component.

Section 6.1, Table 19 (FCS_SSHS_EXT.1) of [ST] states that the TOE's implementation of SSHv2 supports the hashing algorithm hmac-sha1 and hmac-sha2-256 to ensure the integrity of the session. This is consistent with the algorithms specified in the FCS_SSHS_EXT.1.6 requirement.

Guidance Assurance Activities: The evaluator shall also check the guidance documentation to ensure that it contains instructions to the Security Administrator on how to ensure that only the allowed data integrity algorithms are used in SSH connections with the TOE (specifically, that the 'none' MAC algorithm is not allowed).

Section "Secure Communications" in the [AdminGuide] indicates that in the evaluated configuration, SSHv2 must be used instead of SSHv1. The "Enable SSHv2 and Disable SSHv1" section in the [AdminGuide] provides instructions for enabling SSHv2. Section "Hashing Algorithms" in the [AdminGuide] indicates that when SSH version 2 and FIPS mode are enabled, only hmac-sha1 and hmac-sha2-256 are supported for data integrity.

Testing Assurance Activities: Test 1 [conditional, if an HMAC or AEAD_AES_*_GCM algorithm is selected in the ST]: The evaluator shall establish an SSH connection using each of the algorithms, except 'implicit', specified by the requirement. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

Note: To ensure the observed algorithm is used, the evaluator shall ensure a non-aes*- gcm@openssh.com encryption algorithm is negotiated while performing this test.

Test 2 [conditional, if an HMAC or AEAD_AES_*_GCM algorithm is selected in the ST]: The evaluator shall configure an SSH client to only allow a MAC algorithm that is not included the ST selection. The evaluator shall attempt to connect from the SSH client to the TOE and observe that the attempt fails.

Note: To ensure the proposed MAC algorithm is used, the evaluator shall ensure a non-aes*- gcm@openssh.com encryption algorithm is negotiated while performing this test.

Test 1: The evaluator attempted to establish an SSH connection with each of the following SSH transport MAC algorithms: hmac-sha1 and hmac-sha2-256. The evaluator captured packets associated with each of these connection attempts. The evaluator observed that only the hmac-sha1 and hmac-sha2-256 algorithms were able to successfully connect to the TOE.

Test 2: The evaluator attempted to connect to the TOE using HMAC-MD5. The TOE rejects the attempt as expected.

2.2.14.7 NDcPP22e:FCS_SSHS_EXT.1.7

TSS Assurance Activities: The evaluator shall check the TSS to ensure that it lists the supported key exchange algorithms, and that that list corresponds to the list in this component.

Section 6.1, Table 19 (FCS_SSHS_EXT.1) of [ST] states that the TOE's implementation of SSHv2 requires the use of ecdh-sha2-nistp256 and ecdh-sha2-nistp384 key exchanges. This is consistent with the algorithm specified in FCS_SSHS_EXT.1.7.

Guidance Assurance Activities: The evaluator shall also check the guidance documentation to ensure that it contains instructions to the Security Administrator on how to ensure that only the allowed key exchange algorithms are used in SSH connections with the TOE.

Section "Secure Communications" in the [AdminGuide] indicates that the TOE must be running in FIPS mode which is configured using the "fips enable" command. The "Enable SSHv2 and Disable SSHv1" section in the [AdminGuide] describes how to restrict the TOE to use SSHv2. When the TOE is configured with SSHv2 and FIPS mode, the security algorithms and ciphers available on the TOE are restricted.

Section "Key-Exchange" in the [AdminGuide] indicates that SSH key-exchange must be required to use only DH group 14 in the evaluated configuration and provides the instructions to do so using the "ssh key-exchange group" command.

Testing Assurance Activities: Test 1: The evaluator shall configure an SSH client to only allow the diffie-hellman-group1-sha1 key exchange. The evaluator shall attempt to connect from the SSH client to the TOE and observe that the attempt fails.

Test 2: For each allowed key exchange method, the evaluator shall configure an SSH client to only allow that method for key exchange, attempt to connect from the client to the TOE, and observe that the attempt succeeds.

Test 1 - The evaluator attempted to connect to the TOE using Diffie-Hellman-Group1. The TOE rejects the attempt as expected.

Test 2 - The evaluator attempted to establish an SSH connection with each of the following key exchange methods: ecdh-sha2-nistp256 and ecdh-sha2-nistp384. The evaluator captured packets associated with each of these connection attempts. Only the ecdh-sha2-nistp256 and ecdh-sha2-nistp384 algorithms were successful.

2.2.14.8 NDcPP22e:FCS_SSHS_EXT.1.8

TSS Assurance Activities: The evaluator shall check that the TSS specifies the following:

a) Both thresholds are checked by the TOE.

b) Rekeying is performed upon reaching the threshold that is hit first.

Section 6.1, Table 19 (FCS_SSHS_EXT.1) of [ST] states that an SSH connection is rekeyed before 60 minutes of connection time or 1 GB of data traffic, whichever threshold is met first. An audit log is generated to indicate successful rekey.

Guidance Assurance Activities: If one or more thresholds that are checked by the TOE to fulfil the SFR are configurable, then the evaluator shall check that the guidance documentation describes how to configure those thresholds. Either the allowed values are specified in the guidance documentation and must not exceed the limits specified in the SFR (one hour of session time, one gigabyte of transmitted traffic) or the TOE must not accept values beyond the limits specified in the SFR. The evaluator shall check that the guidance documentation describes that the TOE reacts to the first threshold reached.

The “SSH Session Rekey Limits” section in the [AdminGuide] states that the SSH session rekey limits are not configurable. SSH sessions will renew their keys within 60 minutes and 1GB of traffic, whichever limit is reached first.

Testing Assurance Activities: The evaluator needs to perform testing that rekeying is performed according to the description in the TSS. The evaluator shall test both, the time-based threshold and the traffic-based threshold.

For testing of the time-based threshold, the evaluator shall use an SSH client to connect to the TOE and keep the session open until the threshold is reached. The evaluator shall verify that the SSH session has been active longer than the threshold value and shall verify that the TOE initiated a rekey (the method of verification shall be reported by the evaluator).

Testing does not necessarily have to be performed with the threshold configured at the maximum allowed value of one hour of session time but the value used for testing shall not exceed one hour. The evaluator needs to ensure that the rekeying has been initiated by the TOE and not by the SSH client that is connected to the TOE.

For testing of the traffic-based threshold the evaluator shall use the TOE to connect to an SSH client and shall transmit data to and/or receive data from the TOE within the active SSH session until the threshold for data protected by either encryption key is reached. It is acceptable if the rekey occurs before the threshold is reached (e.g. because the traffic is counted according to one of the alternatives given in the Application Note for FCS_SSHS_EXT.1.8).

The evaluator shall verify that more data has been transmitted within the SSH session than the threshold allows and shall verify that the TOE initiated a rekey (the method of verification shall be reported by the evaluator).

Testing does not necessarily have to be performed with the threshold configured at the maximum allowed value of one gigabyte of transferred traffic, but the value used for testing shall not exceed one gigabyte. The evaluator needs to ensure that the rekeying has been initiated by the TOE and not by the SSH client that is connected to the TOE.

If one or more thresholds that are checked by the TOE to fulfil the SFR are configurable, the evaluator needs to verify that the threshold(s) can be configured as described in the guidance documentation and the evaluator needs to test that modification of the thresholds is restricted to Security Administrators (as required by FMT_MOF.1(3)/Functions).

In cases where data transfer threshold could not be reached due to hardware limitations it is acceptable to omit testing of this (SSH rekeying based on data transfer threshold) threshold if both the following conditions are met:

a) An argument is present in the TSS section describing this hardware-based limitation and

b) All hardware components that are the basis of such argument are definitively identified in the ST. For example, if specific Ethernet Controller or WiFi radio chip is the root cause of such limitation, these chips must be identified.

The evaluator then opened an SSH session and monitored the session for a re-key message being sent from the TOE. The evaluator observed that a re-key was sent by the TOE at before 60 minutes. The evaluator then opened a new SSH session and performed actions within the SSH session that caused data transfers. The evaluator observed that the TOE re-keyed before 1GB of data was transferred.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.15 TLS Client Protocol Without Mutual Authentication - per TD0670 & TD0790 (NDcPP22e:FCS_TLSC_EXT.1)

2.2.15.1 NDcPP22e:FCS_TLSC_EXT.1.1

TSS Assurance Activities: The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the ciphersuites supported are specified. The evaluator shall check the TSS to ensure that the ciphersuites specified include those listed for this component.

Section 6.1, Table 19 (FCS_TLSC_EXT.1) of [ST] indicates that the TOE supports TLSv1.2 for secure syslog connections and supports the following ciphersuites:

- TLS_DHE_RSA_WITH_AES_128_CBC_SHA
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA
- TLS_DHE_RSA_WITH_AES_128_CBC_ SHA256
- TLS_DHE_RSA_WITH_AES_256_CBC_ SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384

These ciphersuites are consistent with those identified in the FCS_TLSC_EXT.1 requirement.

Guidance Assurance Activities: The evaluator shall check the guidance documentation to ensure that it contains instructions on configuring the TOE so that TLS conforms to the description in the TSS.

The “Configuring TLS” section in the [AdminGuide] states that in the certified configuration, by default, only TLSv1.2 is enabled. The “ssl client-version” command specifies the TLS protocol version the ASA uses when acting as a client. The [AdminGuide] identifies the options available as TLSv1.2.

The section “Specify the TLS Version” in the [AdminGuide] describes how the TOE can also be configured to support a maximum TLS version of TLSv1.2, and to restrict which TLS ciphersuites will be used by its TLS server and its client by using the ‘ssl cipher’ command. The TLS versions and ciphersuites available are consistent with the requirements in the ST.

Testing Assurance Activities: Test 1: The evaluator shall establish a TLS connection using each of the ciphersuites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an HTTPS session. It is sufficient to observe the successful negotiation of a ciphersuite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic to discern the ciphersuite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

Test 2: The evaluator shall attempt to establish the connection using a server with a server certificate that contains the Server Authentication purpose in the extendedKeyUsage extension and verify that a connection is established. The evaluator will then verify that the client rejects an otherwise valid server certificate that lacks the Server Authentication purpose in the extendedKeyUsage field, and a connection is not established. Ideally, the two certificates should be identical except for the extendedKeyUsage field.

Test 3: The evaluator shall send a server certificate in the TLS connection that does not match the server-selected ciphersuite (for example, send an ECDSA certificate while using the TLS_RSA_WITH_AES_128_CBC_SHA

ciphersuite). The evaluator shall verify that the TOE disconnects after receiving the server's Certificate handshake message.

Test 4: The evaluator shall perform the following 'negative tests':

a) The evaluator shall configure the server to select the TLS_NULL_WITH_NULL_NULL ciphersuite and verify that the client denies the connection.

b) Modify the server's selected ciphersuite in the Server Hello handshake message to be a ciphersuite not presented in the Client Hello handshake message. The evaluator shall verify that the client rejects the connection after receiving the Server Hello.

c) [conditional]: If the TOE presents the Supported Elliptic Curves/Supported Groups Extension the evaluator shall configure the server to perform an ECDHE or DHE key exchange in the TLS connection using a non-supported curve/group (for example P-192) and shall verify that the TOE disconnects after receiving the server's Key Exchange handshake message.

Test 5: The evaluator shall perform the following modifications to the traffic:

a) Change the TLS version selected by the server in the Server Hello to a non-supported TLS version and verify that the client rejects the connection.

b) [conditional]: If using DHE or ECDH, modify the signature block in the Server's Key Exchange handshake message, and verify that the handshake does not finished successfully, and no application data flows. This test does not apply to cipher suites using RSA key exchange. If a TOE only supports RSA key exchange in conjunction with TLS, then this test shall be omitted.

Test 6: The evaluator performs the following 'scrambled message tests':

a) Modify a byte in the Server Finished handshake message and verify that the handshake does not finish successfully and no application data flows.

b) Send a garbled message from the server after the server has issued the ChangeCipherSpec message and verify that the handshake does not finish successfully and no application data flows.

c) Modify at least one byte in the server's nonce in the Server Hello handshake message and verify that the client rejects the Server Key Exchange handshake message (if using a DHE or ECDHE ciphersuite) or that the server denies the client's Finished handshake message.

For the following tests the evaluator configured the test server to require mutual authentication and configured the TOE to establish a TLS session with a test server.

Test 1: The evaluator established a TLS session from the TOE to a test server with the test server configured to accept connections with only one of the claimed ciphersuites at a time. The evaluator used a network sniffer to

capture the TLS session negotiation. The evaluator examined each traffic capture and observed that the expected TLS cipher was negotiated.

Test 2: The evaluator first connected the TOE to a TLS server with a valid certificate. The evaluator observed via packet capture that the connection was successful. Next, the evaluator connected the TOE to a TLS server that had an invalid certificate missing the serverAuth key usage. The packet capture indicated that the connection was rejected by the TOE.

Test 3: The evaluator sent a server certificate in the TLS connection that did not match the server-selected ciphersuite. The evaluator observed via packet capture that the connection was rejected.

Test 4a: The evaluator configured a test server to accept only the TLS_NULL_WITH_NULL_NULL ciphersuite. The evaluator then attempted to establish a TLS session from the TOE to that test server. The packet capture indicated that the TOE rejected the ciphersuite and disconnected the session.

Test 4b: The evaluator configured a test server to send a ciphersuite not presented in the client hello. The evaluator then attempted to establish a TLS session from the TOE to that test server. The packet capture indicated that the TOE rejected the invalid ciphersuite and disconnected the session.

Test 4c: As the TOE supports DHE ciphers the evaluator configured a test server to perform a DHE key exchange in the TLS connection using a non-supported group (dh1096). The evaluator then attempted to establish a TLS session from the TOE to that test server. The packet capture indicated that the TOE rejected the invalid DHE key exchange and disconnected the session.

Test 5a: The evaluator configured a test server to send an invalid TLS version (1.4). The evaluator then attempted to establish a TLS session from the TOE to that test server. The packet capture indicated that the TOE rejected the invalid TLS version and disconnected the session.

Test 5b: As the TOE supports DHE ciphers the evaluator configured a test server to modify the signature block in the Server's Key Exchange handshake message. The evaluator then attempted to establish a TLS session from the TOE to that test server. The packet capture indicated that the TOE rejected the invalid handshake and disconnected the session.

Test 6: The evaluator connected the TOE to a TLS server which modified traffic according to the scenarios identified above for this test case. In each case, the TOE successfully detected the modifications and rejected the connection to the TOE.

2.2.15.2 NDcPP22e:FCS_TLSC_EXT.1.2

TSS Assurance Activities: The evaluator shall ensure that the TSS describes the client's method of establishing all reference identifiers from the administrator/application-configured reference identifier, including which types of reference identifiers are supported (e.g. application-specific Subject Alternative Names) and whether IP addresses and wildcards are supported.

Note that where a TLS channel is being used between components of a distributed TOE for FPT_ITT.1, the requirements to have the reference identifier established by the user are relaxed and the identifier may also be established through a 'Gatekeeper' discovery process. The TSS should describe the discovery process and highlight how the reference identifier is supplied to the 'joining' component. Where the secure channel is being used between components of a distributed TOE for FPT_ITT.1 and the ST author selected attributes from RFC 5280, the evaluator shall ensure the TSS describes which attribute type, or combination of attributes types, are used by the client to match the presented identifier with the configured identifier. The evaluator shall ensure the TSS presents an argument how the attribute type, or combination of attribute types, uniquely identify the remote TOE component; and the evaluator shall verify the attribute type, or combination of attribute types, is sufficient to support unique identification of the maximum supported number of TOE components.

If IP addresses are supported in the CN as reference identifiers, the evaluator shall ensure that the TSS describes the TOE's conversion of the text representation of the IP address in the CN to a binary representation of the IP address in network byte order. The evaluator shall also ensure that the TSS describes whether canonical format (RFC5952 for IPv6, RFC 3986 for IPv4) is enforced.

Section 6.1, Table 19 (FCS_TLSC_EXT.1) in [ST] states that when the TOE acts as a TLS client, the administrators can specify the reference-identity using the following command: 'crypto ca reference-identity reference-identity-name'. This is followed by one or more of the values (where cn-id would be used to specify the FQDN or DN): cn-id value, dns-id value, uri-id value. To configure the syslog server certification verification, the following syntax should be used:

```
'logging host interface_name syslog_ip [tcp/port | udp/port] [format emblem] [secure
[reference-identity reference_identity_name]] [permit-hostdown]'.
```

When ASA validates a certificate against a reference-identity configuration, the uri-id is matched literally and the cn-id and dns-id support wildcards.

The TOE is not distributed and does not support IP addresses as reference identifiers.

Guidance Assurance Activities: The evaluator shall ensure that the operational guidance describes all supported identifiers, explicitly states whether the TOE supports the SAN extension or not, and includes detailed instructions on how to configure the reference identifier(s) used to check the identity of peer(s). If the identifier scheme implemented by the TOE includes support for IP addresses, the evaluator shall ensure that the operational guidance provides a set of warnings and/or CA policy recommendations that would result in secure TOE use.

Where the secure channel is being used between components of a distributed TOE for FPT_ITT.1, the SFR selects attributes from RFC 5280, and FCO_CPC_EXT.1.2 selects 'no channel'; the evaluator shall verify the guidance provides instructions for establishing unique reference identifiers based on RFC5280 attributes.

The “Configure Reference Identifier” section of the [AdminGuide] states that the **crypto ca reference-identity** command in configuration mode is used to configure a reference-identity object to define certificate matching rules for TLS connections.

The **crypto ca reference-identity** command in global configuration mode is used to place the ASA in ca-reference-identity mode. The following reference-ids can then be added. Multiple reference-ids of any type may be added. The no form of each command is used to remove reference-ids.

[**no**] **cn-id** *value* - Common Name (CN) where value matches the overall form of a domain name.

[**no**] **dns-id** *value* - a subjectAltName entry of type dNSName. This is a DNS domain name.

[**no**] **uri-id** *value* - a subjectAltName entry of type uniformResourceIdentifier. A URI-ID identifier must contain the DNS domain name, not the IP address, and not just the hostname.

When the ASA is acting as a TLS client, it supports rules for verification of an application server's identity as defined in RFC 6125. Reference identities are configured on the ASA, to be compared to the identity presented in a server certificate during connection establishment. These identifiers are specific instances of identifier types also specified in RFC 6125.

The TOE does not claim support for IP addresses as reference identifiers.

The TOE is not distributed.

Testing Assurance Activities: Note that the following tests are marked conditional and are applicable under the following conditions:

a) For TLS-based trusted channel communications according to FTP_ITC.1 where RFC 6125 is selected, tests 1-6 are applicable.

or

b) For TLS-based trusted path communications according to FTP_TRP where RFC 6125 is selected, tests 1-6 are applicable

or

c) For TLS-based trusted path communications according to FPT_ITT.1 where RFC 6125 is selected, tests 1-6 are applicable. Where RFC 5280 is selected, only test 7 is applicable.

Note that for some tests additional conditions apply.

IP addresses are binary values that must be converted to a textual representation when presented in the CN of a certificate. When testing IP addresses in the CN, the evaluator shall follow the following formatting rules:

- IPv4: The CN contains a single address that is represented a 32-bit numeric address (IPv4) is written in decimal as four numbers that range from 0-255 separated by periods as specified in RFC 3986.

- IPv6: The CN contains a single IPv6 address that is represented as eight colon separated groups of four lowercase hexadecimal digits, each group representing 16 bits as specified in RFC 4291. Note: Shortened addresses, suppressed zeros, and embedded IPv4 addresses are not tested.

The evaluator shall configure the reference identifier according to the AGD guidance and perform the following tests during a TLS connection:

a) Test 1 [conditional]: The evaluator shall present a server certificate that contains a CN that does not match the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each identifier type (e.g. IPv4, IPv6, FQDN) supported in the CN. When testing IPv4 or IPv6 addresses, the evaluator shall modify a single decimal or hexadecimal digit in the CN.

Remark: Some systems might require the presence of the SAN extension. In this case the connection would still fail but for the reason of the missing SAN extension instead of the mismatch of CN and reference identifier. Both reasons are acceptable to pass Test 1.

b) Test 2 [conditional]: The evaluator shall present a server certificate that contains a CN that matches the reference identifier, contains the SAN extension, but does not contain an identifier in the SAN that matches the reference identifier. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each supported SAN type (e.g. IPv4, IPv6, FQDN, URI). When testing IPv4 or IPv6 addresses, the evaluator shall modify a single decimal or hexadecimal digit in the SAN.

c) Test 3 [conditional]: If the TOE does not mandate the presence of the SAN extension, the evaluator shall present a server certificate that contains a CN that matches the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection succeeds. The evaluator shall repeat this test for each identifier type (e.g. IPv4, IPv6, FQDN) supported in the CN. If the TOE does mandate the presence of the SAN extension, this Test shall be omitted.

d) Test 4 [conditional]: The evaluator shall present a server certificate that contains a CN that does not match the reference identifier but does contain an identifier in the SAN that matches. The evaluator shall verify that the connection succeeds. The evaluator shall repeat this test for each supported SAN type (e.g. IPv4, IPv6, FQDN, SRV).

e) Test 5 [conditional]: The evaluator shall perform the following wildcard tests with each supported type of reference identifier that includes a DNS name (i.e. CN-ID with DNS, DNS-ID, SRV-ID, URI-ID):

1) [conditional]: The evaluator shall present a server certificate containing a wildcard that is not in the left- most label of the presented identifier (e.g. foo.*.example.com) and verify that the connection fails.

2) [conditional]: The evaluator shall present a server certificate containing a wildcard in the left-most label (e.g. *.example.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.example.com) and verify that the connection succeeds if wildcards are supported or fails if wildcards are not supported. The evaluator shall configure the reference identifier without a left-most label as in the certificate (e.g. example.com) and verify that the connection fails. The evaluator shall configure the reference identifier with two left-most labels (e.g. bar.foo.example.com) and verify that the connection fails. (Remark: Support for wildcards was always intended to be optional. It is sufficient to state that the TOE does not support wildcards and observe rejected connection attempts to satisfy corresponding assurance activities.)

f) Objective: The objective of this test is to ensure the TOE is able to differentiate between IP address identifiers that are not allowed to contain wildcards and other types of identifiers that may contain wildcards.

Test 6: [conditional]If IP address identifiers are supported in the SAN or CN, the evaluator shall present a server certificate that contains a CN that matches the reference identifier, except one of the groups has been replaced with a wildcard asterisk (*) (e.g. CN=*.168.0.1 when connecting to 192.168.1.20, CN=2001:0DB8:0000:0000:0008:0800:200C:* when connecting to 2001:0DB8:0000:0000:0008:0800:200C:417A). The certificate shall not contain the SAN extension. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each supported IP address version (e.g. IPv4, IPv6).

This negative test corresponds to the following section of the Application Note 64/105: 'The exception being, the use of wildcards is not supported when using IP address as the reference identifier.'

Remark: Some systems might require the presence of the SAN extension. In this case the connection would still fail but for the reason of the missing SAN extension instead of the mismatch of CN and reference identifier. Both reasons are acceptable to pass Test 6.

(TD0790 applied, supersedes TD0670)

Test 7 [conditional]: If the secure channel is used for FPT_ITT, and RFC 5280 is selected, the evaluator shall perform the following tests. Note, when multiple attribute types are selected in the SFR (e.g. when multiple attribute types are combined to form the unique identifier), the evaluator modifies each attribute type in accordance with the matching criteria described in the TSS (e.g. creating a mismatch of one attribute type at a time while other attribute types contain values that will match a portion of the reference identifier):

1) The evaluator shall present a server certificate that does not contain an identifier in the Subject (DN) attribute type(s) that matches the reference identifier. The evaluator shall verify that the connection fails.

2) The evaluator shall present a server certificate that contains a valid identifier as an attribute type other than the expected attribute type (e.g. if the TOE is configured to expect id-atserialNumber=correct_identifier, the certificate could instead include id-at-name=correct_identifier), and does not contain the SAN extension. The evaluator shall verify that the connection fails. Remark: Some systems might require the presence of the SAN extension. In this case the connection would still fail but for the reason of the missing SAN extension instead of the mismatch of CN and reference identifier. Both reasons are acceptable to pass this test.

3) The evaluator shall present a server certificate that contains a Subject attribute type that matches the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection succeeds.

4) The evaluator shall confirm that all use of wildcards results in connection failure regardless of whether the wildcards are used in the left or right side of the presented identifier. (Remark: Use of wildcards is not addressed within RFC 5280.)

The test results show TOE support for each reference identifier type. The TOE can be configured to identify the following:

- a CN-id value that must be in a certificate's CN field,
- a DNS-id value that must be in the certificate's SAN DNS field or CN field, or
- a URI-id value that must be in the certificate's SAN URI field.

The TOE accepts certificates using wildcards for DNS names that may be contained either in a SAN DNS field or a CN field. The TOE does not match wildcards in certificates to against URI-id values.

Test 1: The evaluator established a TLS session from the TOE targeting a server using a valid certificate with a CN matching the domain name used by the client. Using a network sniffer to capture the TLS session negotiation, the evaluator examined the traffic capture and observed a successful connection. The evaluator then established a TLS session from the TOE targeting a server using a server certificate that does not contain an identifier in either the Subject Alternative Name (SAN) or Common Name (CN) that matches the reference identifier. Using a network sniffer to capture the TLS session negotiation, the evaluator examined the traffic capture and observed that the TLS session was not negotiated successfully.

Test 2: The evaluator established a TLS session from the TOE targeting a server using a server certificate that contains a CN that matches the reference identifier, contains the SAN extension, but does not contain an identifier in the SAN that matches the reference identifier. Using a network sniffer to capture the TLS session negotiation, the evaluator examined the traffic capture and observed that the TLS session was not negotiated successfully.

Test 3: The evaluator established a TLS session from the TOE targeting a server using a server certificate that contains a CN that matches the reference identifier and does not contain the SAN extension. Using a network sniffer to capture the TLS session negotiation, the evaluator examined the traffic capture and observed that the TLS session was negotiated successfully.

Test 4: The evaluator established a TLS session from the TOE targeting a server using a server certificate that contains a CN that does not match the reference identifier but does contain an identifier in the SAN that matches. Using a network sniffer to capture the TLS session negotiation, the evaluator examined the traffic capture and observed that the TLS session was negotiated successfully.

Test 5: The evaluator tested hostname wildcards by configuring an expected DNS on the TOE with the test server's certificates configured with wildcard DNS names. The TOE does not support the use of wildcards in certificates. The TOE rejected all certificates that included hostname wildcards.

Certificate Contents	**Host ID**	**Expected Result**
CN=bar.*.example.com	bar.foo.example.com	No Connection
SAN=bar.*.example.com	bar.foo.example.com	No Connection
CN=*.example.com	bar.foo.example.com	No Connection
SAN=*.example.com	bar.foo.example.com	No Connection
CN=*.example.com	foo.example.com	Successful Connection
SAN=*.example.com	foo.example.com	Successful Connection
CN=*.com	example.com	No Connection
SAN=*.com	example.com	No Connection

Test 6: The TOE does not support IP addresses in the CN.

Test 7: The TOE does not support FPT_ITT.1

2.2.15.3 NDcPP22e:FCS_TLSC_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall demonstrate that using an invalid certificate results in the function failing as follows:

Test 1: Using the administrative guidance, the evaluator shall load a CA certificate or certificates needed to validate the presented certificate used to authenticate an external entity and demonstrate that the function succeeds and a trusted channel can be established.

Test 2: The evaluator shall then change the presented certificate(s) so that validation fails and show that the certificate is not automatically accepted. The evaluator shall repeat this test to cover the selected types of failure defined in the SFR (i.e. the selected ones from failed matching of the reference identifier, failed validation of the certificate path, failed validation of the expiration date, failed determination of the revocation status). The evaluator performs the action indicated in the SFR selection observing the TSF resulting in the expected state for the trusted channel (e.g. trusted channel was established) covering the types of failure for which an override mechanism is defined.

Test 3: The purpose of this test to verify that only selected certificate validation failures could be administratively overridden. If any override mechanism is defined for failed certificate validation, the evaluator shall configure a new presented certificate that does not contain a valid entry in one of the mandatory fields or parameters (e.g. inappropriate value in extendedKeyUsage field) but is otherwise valid and signed by a trusted CA. The evaluator shall confirm that the certificate validation fails (i.e. certificate is rejected), and there is no administrative override available to accept such certificate.

Test 1 and 2 were performed as part of testing of FIA_X509_EXT.1.1/Rev Test 1. Test 3 is not applicable as the TOE does not offer administrative override of certificate validation failures.

2.2.15.4 NDcPP22e:FCS_TLSC_EXT.1.4

TSS Assurance Activities: The evaluator shall verify that TSS describes the Supported Elliptic Curves/Supported Groups Extension and whether the required behavior is performed by default or may be configured.

Section 6.1, Table 19 (FCS_TLSC_EXT.1) of [ST] states that NIST curves (secp256r1, secp384r1 and secp521r1) are supported by default but mutual authentication must be configured with the client-side X.509v3 certificate.

Guidance Assurance Activities: If the TSS indicates that the Supported Elliptic Curves/Supported Groups Extension must be configured to meet the requirement, the evaluator shall verify that AGD guidance includes configuration of the Supported Elliptic Curves/Supported Groups Extension.

The TSS indicates that the NIST ECC curves secp256r1, secp384r1, secp521r1 Supported Elliptic Curves/Supported Groups Extension are configured by default. Therefore, the [AdminGuide] does not need to address configuration of the Supported Elliptic Curves/Supported Groups Extension.

Testing Assurance Activities: Test 1 [conditional]: If the TOE presents the Supported Elliptic Curves/Supported Groups Extension, the evaluator shall configure the server to perform ECDHE or DHE (as applicable) key exchange using each of the TOE's supported curves and/or groups. The evaluator shall verify that the TOE successfully connects to the server.

Test 1: For ECDHE key exchanges, the evaluator attempted to establish a TLS session between the TOE and a test server configured to allow only one key exchange method. The evaluator observed that the TOE was able to connect with the test server using the following key exchange methods.

- ECDHE w/ P-256 curve
- ECDHE w/ P-384 curve
- ECDHE w/ P-521 curve

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.16 TLS Client Support for Mutual Authentication - per TD0670 (NDcPP22e:FCS_TLSC_EXT.2)

2.2.16.1 NDcPP22e:FCS_TLSC_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure that the TSS description required per FIA_X509_EXT.2.1 includes the use of client-side certificates for TLS mutual authentication.

Section 6.1, Table 19 (FCS_TLSC_EXT.2) of [ST] states that NIST curves are supported by default but mutual authentication must be configured with the client-side X.509v3 certificate.

Component Guidance Assurance Activities: If the TSS indicates that mutual authentication using X.509v3 certificates is used, the evaluator shall verify that the AGD guidance includes instructions for configuring the client-side certificates for TLS mutual authentication.

The “Specify the TLS Server and Client Certificates” section of the [AdminGuide] states that the certificate is configured by the “ssl trust-point” command specifying the certificate and interface**.** If the server requests a client

certificate from the ASA for authentication, the ASA will send whatever client identity certificate is configured for that interface.

The “Create Trustpoint and Generate Certificate Signing Request (CSR)” section of the [AdminGuide] describes how to create a trustpoint and import a device/client certificate into that trustpoint.

Component Testing Assurance Activities: For all tests in this chapter the TLS server used for testing of the TOE shall be configured to require mutual authentication.

Test 1: The evaluator shall establish a connection to a peer server that is configured for mutual authentication (i.e. sends a server Certificate Request (type 13) message). The evaluator observes that the TOE TLS client sends both client Certificate (type 11) and client Certificate Verify (type 15) messages during its negotiation of a TLS channel and that Application Data is sent.

In addition, all other testing in FCS_TLSC_EXT.1 and FIA_X509_EXT.* must be performed as per the requirements.

The evaluator performed all TLS client tests such that the test server used for testing of FCS_TLSC_EXT.1.1 Test 1 required mutual authentication. In all of the test cases the TOE responded in accordance with the AA and Security Target claims.

Test 1: The evaluator configured a peer server to send a Certificate Request message during TLS negotiation and observed that the TOE did send the client Certificate and Client Verify messages in response. The evaluator also observed application data flowing after FINISH messages by both peers.

2.2.17 TLS Server Protocol Without Mutual Authentication - per TD0635 (NDcPP22e:FCS_TLSS_EXT.1)

2.2.17.1 NDcPP22e:FCS_TLSS_EXT.1.1

TSS Assurance Activities: The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the ciphersuites supported are specified. The evaluator shall check the TSS to ensure that the ciphersuites specified are identical to those listed for this component.

Section 6.1, Table 19 (FCS_TLSS_EXT.1) of [ST] states that the TOE implements HTTP over TLS (HTTPS) to support remote administration using TLS v1.2 connections with any of the following ciphersuites:

- TLS_DHE_RSA_WITH_AES_128_CBC_SHA
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA
- TLS_DHE_RSA_WITH_AES_128_CBC_ SHA256
- TLS_DHE_RSA_WITH_AES_256_CBC_ SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384

These ciphersuites are consistent with those identified in the FCS_TLSS_EXT.1 requirement.

Guidance Assurance Activities: The evaluator shall check the guidance documentation to ensure that it contains instructions on configuring the TOE so that TLS conforms to the description in the TSS (for instance, the set of ciphersuites advertised by the TOE may have to be restricted to meet the requirements).

The “Configuring TLS” section in the [AdminGuide] states that in the certified configuration, by default, only TLSv1.2 is enabled. The “ssl client-version” command specifies the TLS protocol version the ASA uses when acting as a client. The [AdminGuide] identifies the options available as TLSv1.2.

The section “Specify the TLS Version” in the [AdminGuide] describes how the TOE can also be configured to support a maximum TLS version of TLSv1.2, and to restrict which TLS ciphersuites will be used by its TLS server and its client by using the ‘ssl cipher’ command. The TLS versions and ciphersuites available are consistent with the requirements in the ST.

Testing Assurance Activities: Test 1: The evaluator shall establish a TLS connection using each of the ciphersuites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an HTTPS session. It is sufficient to observe the successful negotiation of a ciphersuite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic to discern the ciphersuite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

Test 2: The evaluator shall send a Client Hello to the server with a list of ciphersuites that does not contain any of the ciphersuites in the server's ST and verify that the server denies the connection. Additionally, the evaluator shall send a Client Hello to the server containing only the TLS_NULL_WITH_NULL_NULL ciphersuite and verify that the server denies the connection.

Test 3: The evaluator shall perform the following modifications to the traffic:

a) Modify a byte in the Client Finished handshake message, and verify that the server rejects the connection and does not send any application data.

b) (Test Intent: The intent of this test is to ensure that the server's TLS implementation immediately makes use of the key exchange and authentication algorithms to: a) Correctly encrypt (D)TLS Finished message and b) Encrypt every (D)TLS message after session keys are negotiated.)

The evaluator shall use one of the claimed ciphersuites to complete a successful handshake and observe transmission of properly encrypted application data. The evaluator shall verify that no Alert with alert level Fatal (2) messages were sent.

The evaluator shall verify that the Finished message (Content type hexadecimal 16 and handshake message type hexadecimal 14) is sent immediately after the server's ChangeCipherSpec (Content type hexadecimal 14) message. The evaluator shall examine the Finished message (encrypted example in hexadecimal of a TLS record containing a Finished message, 16 03 03 00 40 11 22 33 44 55...) and confirm that it does not contain unencrypted data (unencrypted example in hexadecimal of a TLS record containing a Finished message, 16 03 03 00 40 14 00 00 0c...), by verifying that the first byte of the encrypted Finished message does not equal hexadecimal 14 for at least one of three test messages. There is a chance that an encrypted Finished message contains a hexadecimal value of

'14' at the position where a plaintext Finished message would contain the message type code '14'. If the observed Finished message contains a hexadecimal value of '14' at the position where the plaintext Finished message would contain the message type code, the test shall be repeated three times in total. In case the value of '14' can be observed in all three tests it can be assumed that the Finished message has indeed been sent in plaintext and the test has to be regarded as 'failed'. Otherwise it has to be assumed that the observation of the value '14' has been due to chance and that the Finished message has indeed been sent encrypted. In that latter case the test shall be regarded as 'passed'.

Test 1: The evaluator attempted to connect to the TOE using each of the claimed ciphersuites. A packet capture was obtained for each connection attempt. The evaluator confirmed that each connection was successful.

Test 2: The evaluator sent a Client Hello to the server with a list of ciphersuites that does not contain any of the ciphersuites in the server's ST and verified that the server denied all such connection attempts. Additionally, the evaluator sent a Client Hello to the server containing only the TLS_NULL_WITH_NULL_NULL ciphersuite and verified that the server denied the connection.

Test 3: (a, b): The evaluator made connection attempts from a client to the TOE. The client implementation of the TLS protocol was modified as stated in parts (a) and (b) of the assurance activity. The evaluator observed in packet captures that the connections are rejected for each scenario. Also, the evaluator established a TLS connection from a test server to the TOE, while capturing packets associated with that connection. The evaluator located the ChangeCipherSpec message (Content type hexadecimal 14) in the packet capture and found it to be followed by an EncryptedHandshakeMessage (Content type hexadecimal 16). The evaluator examined the payload of the EncryptedHandshakeMessage to determine if it contained a cleartext or encrypted version of the required Finished message. A Cleartext version would begin with a Content type hexadecimal value of 14, while an encrypted version would (for at least one of three test messages) not contain a Content type hexadecimal value of 14.

2.2.17.2 NDcPP22e:FCS_TLSS_EXT.1.2

TSS Assurance Activities: The evaluator shall verify that the TSS contains a description of how the TOE technically prevents the use of old SSL and TLS versions.

Section 6.1, Table 19 (FCS_TLSS_EXT.1) of [ST] states that the TOE support TLSv1.2. It also indicates that connections not supporting the configured TLS version will not be established.

Guidance Assurance Activities: The evaluator shall verify that any configuration necessary to meet the requirement must be contained in the AGD guidance.

The “Configuring TLS” section in the [AdminGuide] states that in the certified configuration, by default, only TLSv1.2 is enabled. The “ssl client-version” command specifies the TLS protocol version the ASA uses when acting as a client. The [AdminGuide] identifies the options available as TLSv1.2.

The section “Specify the TLS Version” in the [AdminGuide] describes how the TOE can also be configured to support a maximum TLS version of TLSv1.2, and to restrict which TLS ciphersuites will be used by its TLS server and

its client by using the 'ssl cipher' command. The TLS versions and ciphersuites available are consistent with the requirements in the ST.

Testing Assurance Activities: The evaluator shall send a Client Hello requesting a connection for all mandatory and selected protocol versions in the SFR (e.g. by enumeration of protocol versions in a test client) and verify that the server denies the connection for each attempt.

The evaluator attempted to establish a TLS session on the TOE with each of the unsupported versions of SSL and TLS. The evaluator used a network sniffer to capture the session negotiation and observed that the expected protocol and version were offered and rejected during negotiation.

2.2.17.3 NDcPP22e:FCS_TLSS_EXT.1.3

TSS Assurance Activities: If using ECDHE and/or DHE ciphers, the evaluator shall verify that the TSS lists all EC Diffie-Hellman curves and/or Diffie-Hellman groups used in the key establishment by the TOE when acting as a TLS Server. For example, if the TOE supports TLS_DHE_RSA_WITH_AES_128_CBC_SHA cipher and Diffie-Hellman parameters with size 2048 bits, then list Diffie-Hellman Group 14.

Section 6.1, Table 19 (FCS_TLSS_EXT.1) of [ST] states that the key agreement parameters of the server key exchange message are specified in the RFC 5246 (section 7.4.3) for TLSv1.2 is not supported in the evaluated configuration. The TOE performs key establishment for TLS using Diffie-Hellman parameters with size 2048 bits and 3072 bits and ECDHE curves - secp256r1, secp384r1 and secp521r1 [DH group 14, 15, 19, 20 and 24].

Guidance Assurance Activities: The evaluator shall verify that any configuration necessary to meet the requirement must be contained in the AGD guidance.

The "Configuring TLS" section of the [AdminGuide] states that only Diffie-Hellman Group 14 (2048 bits) and Diffie-Hellman Group 15 (3072 bits) should be supported for TLS. The syntax is: ssl dh-group [*group14* | *group15*]. This is consistent with the requirement which only allows Diffie-Hellman parameters of size 2048 bits and 3072 bits. This section also indicates that the TOE supports key agreement using NIST ECC curves secp256r1, secp384r1, and secp521r1.

Testing Assurance Activities: Test 1: [conditional] If ECDHE ciphersuites are supported:

a) The evaluator shall repeat this test for each supported elliptic curve. The evaluator shall attempt a connection using a supported ECDHE ciphersuite and a single supported elliptic curve specified in the Elliptic Curves Extension. The Evaluator shall verify (though a packet capture or instrumented client) that the TOE selects the same curve in the Server Key Exchange message and successfully establishes the connection.

b) The evaluator shall attempt a connection using a supported ECDHE ciphersuite and a single unsupported elliptic curve (e.g. secp192r1 (0x13)) specified in RFC4492, chap. 5.1.1. The evaluator shall verify that the TOE does not send a Server Hello message and the connection is not successfully established.

Test 2: [conditional] If DHE ciphersuites are supported, the evaluator shall repeat the following test for each supported parameter size. If any configuration is necessary, the evaluator shall configure the TOE to use a

supported Diffie-Hellman parameter size. The evaluator shall attempt a connection using a supported DHE ciphersuite. The evaluator shall verify (through a packet capture or instrumented client) that the TOE sends a Server Key Exchange Message where p Length is consistent with the message are the ones configured Diffie-Hellman parameter size(s).

Test 3: [conditional] If RSA key establishment ciphersuites are supported, the evaluator shall repeat this test for each RSA key establishment key size. If any configuration is necessary, the evaluator shall configure the TOE to perform RSA key establishment using a supported key size (e.g. by loading a certificate with the appropriate key size). The evaluator shall attempt a connection using a supported RSA key establishment ciphersuite. The evaluator shall verify (through a packet capture or instrumented client) that the TOE sends a certificate whose modulus is consistent with the configured RSA key size.

Test 1: The evaluator attempted to establish a TLS session with the TOE when the evaluator's TLS client specified only one key exchange method in the Client Hello. The evaluator observed that the TOE selected the same curve as offered by the client for the following key exchange methods.

- ECDHE w/ P-256 curve
- ECDHE w/ P-384 curve
- ECDHE w/ P-521 curve

The evaluator also attempted a connection using ECDHE w/ P-192 curve and observed the TOE rejected the connection attempt.

Test 2: The evaluator attempted to establish a TLS session with the TOE when the evaluator's TLS client specified only one key exchange method in the Client Hello. The evaluator observed that the TOE selected the same curve as offered by the client for the following key exchange methods.

- DHE w/ parameter size 2048-bit
- DHE w/ parameter size 3072-bit

Test 3: Not applicable. The TOE does not support ciphersuites that use RSA key exchange.

2.2.17.4 NDcPP22e:FCS_TLSS_EXT.1.4

TSS Assurance Activities: The evaluator shall verify that the TSS describes if session resumption based on session IDs is supported (RFC 4346 and/or RFC 5246) and/or if session resumption based on session tickets is supported (RFC 5077).

If session tickets are supported, the evaluator shall verify that the TSS describes that the session tickets are encrypted using symmetric algorithms consistent with FCS_COP.1/DataEncryption. The evaluator shall verify that the TSS identifies the key lengths and algorithms used to protect session tickets.

If session tickets are supported, the evaluator shall verify that the TSS describes that session tickets adhere to the structural format provided in section 4 of RFC 5077 and if not, a justification shall be given of the actual session ticket format.

Section 6.1, Table 19 (FCS_TLSS_EXT.1) in [ST} states that TLS session resumption is supported for the TLS connections of the TOE based on session IDs according to RFC 5246 (TLS1.2). The TOE tracks the negotiated sessions based on session IDs and when the TLS client reconnects to the server based on session IDs, it can look up the session keys to resume the encrypted session.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: Test Objective: To demonstrate that the TOE will not resume a session for which the client failed to complete the handshake (independent of TOE support for session resumption).

Test 1 [conditional]: If the TOE does not support session resumption based on session IDs according to RFC4346 (TLS1.1) or RFC5246 (TLS1.2) or session tickets according to RFC5077, the evaluator shall perform the following test:

a) The client sends a Client Hello with a zero-length session identifier and with a SessionTicket extension containing a zero-length ticket.

b) The client verifies the server does not send a NewSessionTicket handshake message (at any point in the handshake).

c) The client verifies the Server Hello message contains a zero-length session identifier or passes the following steps: Note: The following steps are only performed if the ServerHello message contains a non-zero length SessionID.

d) The client completes the TLS handshake and captures the SessionID from the ServerHello.

e) The client sends a ClientHello containing the SessionID captured in step d). This can be done by keeping the TLS session in step d) open or start a new TLS session using the SessionID captured in step d).

f) The client verifies the TOE (1) implicitly rejects the SessionID by sending a ServerHello containing a different SessionID and by performing a full handshake (as shown in Figure 1 of RFC 4346 or RFC 5246), or (2) terminates the connection in some way that prevents the flow of application data.

Test 2 [conditional]: If the TOE supports session resumption using session IDs according to RFC4346 (TLS1.1) or RFC5246 (TLS1.2), the evaluator shall carry out the following steps (note that for each of these tests, it is not necessary to perform the test case for each supported version of TLS):

a) The evaluator shall conduct a successful handshake and capture the TOE-generated session ID in the Server Hello message. The evaluator shall then initiate a new TLS connection and send the previously captured session ID to show that the TOE resumed the previous session by responding with ServerHello containing the same SessionID immediately followed by ChangeCipherSpec and Finished messages (as shown in Figure 2 of RFC 4346 or RFC 5246).

b) The evaluator shall initiate a handshake and capture the TOE-generated session ID in the Server Hello message. The evaluator shall then, within the same handshake, generate or force an unencrypted fatal Alert message

immediately before the client would otherwise send its ChangeCipherSpec message thereby disrupting the handshake. The evaluator shall then initiate a new Client Hello using the previously captured session ID, and verify that the server (1) implicitly rejects the session ID by sending a ServerHello containing a different SessionID and performing a full handshake (as shown in figure 1 of RFC 4346 or RFC 5246), or (2) terminates the connection in some way that prevents the flow of application data.

Test 3 [conditional]: If the TOE supports session tickets according to RFC5077, the evaluator shall carry out the following steps (note that for each of these tests, it is not necessary to perform the test case for each supported version of TLS):

a) The evaluator shall permit a successful TLS handshake to occur in which a session ticket is exchanged with the non-TOE client. The evaluator shall then attempt to correctly reuse the previous session by sending the session ticket in the ClientHello. The evaluator shall confirm that the TOE responds with a ServerHello with an empty SessionTicket extension, NewSessionTicket, ChangeCipherSpec and Finished messages (as seen in figure 2 of RFC 5077).

b) The evaluator shall permit a successful TLS handshake to occur in which a session ticket is exchanged with the non-TOE client. The evaluator will then modify the session ticket and send it as part of a new Client Hello message. The evaluator shall confirm that the TOE either (1) implicitly rejects the session ticket by performing a full handshake (as shown in figure 3 or 4 of RFC 5077), or (2) terminates the connection in some way that prevents the flow of application data.

Test 1: Not applicable. The TOE claims to support session resumption using session ID values.

Test 2a: The evaluator first attempted a successful TLS negotiation capturing the TOE-generated session ID value from the server hello message. The evaluator initiated a new TLS connection sending the session ID value obtained during the successful negotiation. The evaluator observed that the TOE resumed the first TLS session because the TOE responded with a ServerHello message containing the original Session ID value followed by ChangeCipherSpec and Finished messages.

Test 2b: The evaluator attempted to open a TLS connection to the TOE where the evaluator's client recorded the session_id from a failed TLS negotiation. The evaluator observed the TOE reject the connection.

Test 3: Not applicable. The TOE claims to support session resumption using session ID values.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.3 User data protection (FDP)

2.3.1 Full Residual Information Protection (STFFW14E:FDP_RIP.2)

2.3.1.1 STFFW14E:FDP_RIP.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 'Resources' in the context of this requirement are network packets being sent through (as opposed to 'to', as is the case when a security administrator connects to the TOE) the TOE. The concern is that once a network packet is sent, the buffer or memory area used by the packet still contains data from that packet, and that if that buffer is re-used, those data might remain and make their way into a new packet. The evaluator shall check to ensure that the TSS describes packet processing to the extent that they can determine that no data will be reused when processing network packets. The evaluator shall ensure that this description at a minimum describes how the previous data are zeroized/overwritten, and at what point in the buffer processing this occurs.

Section 6.1, Table 19 (FDP_RIP.2[FW]) in [ST] states that the TOE ensures that packets transmitted from the TOE do not contain residual information from previous packets. Packets that are not the required length use zeros for padding. Residual data is never transmitted from the TOE. Packet handling within memory buffers ensures new packets cannot contain portions of previous packets. This applies to data plane traffic and even administrative session traffic.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.4 Firewall (FFW)

2.4.1 Stateful Traffic Filtering (STFFW14E:FFW_RUL_EXT.1)

2.4.1.1 STFFW14E:FFW_RUL_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.1.2 STFFW14E:FFW_RUL_EXT.1.2

TSS Assurance Activities: The evaluator shall verify that the TSS describes a stateful packet filtering policy and the following attributes are identified as being configurable within stateful traffic filtering rules for the associated protocols:

- ICMPv4

o Type

o Code

- ICMPv6

o Type

o Code

- IPv4

o Source address

o Destination Address

o Transport Layer Protocol

- IPv6

o Source address

o Destination Address

o Transport Layer Protocol and where defined by the ST author, Extension Header Type, Extension Header Fields

- TCP

o Source Port

o Destination Port

- UDP

o Source Port

o Destination Port

The evaluator shall verify that each rule can identify the following actions: permit or drop with the option to log the operation. The evaluator shall verify that the TSS identifies all interface types subject to the stateful packet filtering policy and explains how rules are associated with distinct network interfaces.

Section 6.1, Table 19 of [ST] provides the following:

FFW_RUL_EXT.1.1[FW] states that the TOE provides stateful traffic filtering of IPv4 and IPv6 network traffic. Administratively-defined traffic filter rules (access-lists) can be applied to any interface to filter traffic based on IP

parameters including source and destination address, transport layer protocol, type and code, TCP and UDP port numbers. The TOE allows establishment of communications between remote endpoints, and tracks the state of each session (e.g. initiating, established, and tear-down), and will clear established sessions after proper tear-down is completed as defined by each protocol, or when session timeouts are reached.

The proper session establishment and termination followed by the TOE is as defined in the following RFCs:

- RFC 792 (ICMPv4)
- RFC 4443 (ICMPv6)
- RFC 791 (IPv4)
- RFC 2460 (IPv6)
- TCP, RFC 793, section 2.7 Connection Establishment and Clearing
- UDP, RFC 768 (not applicable, UDP is a “stateless” protocol)

FFW_RUL_EXT.1.2[FW] states that the TOE supports filtering of the following protocols and enforces proper session establishment, management, and termination as defined in each protocol’s RFC including proper use of:

- Addresses, type of service, fragmentation data, size and padding, and IP options including loose source routing, strict source routing, and record route as defined in RFC 791 (IPv4), and RFC 2460 (IPv6);
- Source and destination addresses, Port numbers, sequence and acknowledgement numbers, size and padding, and control bits such as SYN, ACK, FIN, and RST as defined in RFC 793 (TCP);
- Source and destination addresses, Port numbers, and length as defined in RFC 768 (UDP); and
- Session identifiers, sequence numbers, types, and codes as defined in RFC 792 (ICMPv4), and RFC 4443 (ICMPv6).

FFW_RUL_EXT.1.3[FW]/FFW_RUL_EXT.1.4[FW] states that each traffic flow control rule on the TOE is defined as either a “permit” rule, or a “deny” rule, and any rule can also contain the keyword “log” which will cause a log message to be generated when a new session is established because it matched the rule. The TOE can be configured to generate a log message for the session establishment of any permitted or denied traffic (in this case, attempt to establish a session).

Access Control Lists (ACLs) are only enforced after they’ve been applied to a network interface. Any network interface can have an ACL applied to it with the “access-group” command, e.g. “access-group sample-acl in interface outside”. Interfaces can be referred to by their identifier (e.g. GigabitEthernet 0/1), or by a name if named using the “nameif” command.

The interface types that can be assigned to an access-group are:

Physical interfaces

- Ethernet
- GigabitEthernet
- TenGigabitEthernet

Management

- Port-channel interfaces (designated by a port-channel number)
- Subinterface (designated by the subinterface number)

Guidance Assurance Activities: The evaluators shall verify that the guidance documentation identifies the following attributes as being configurable within stateful traffic filtering rules for the associated protocols:

- ICMPv4

o Type

o Code

- ICMPv6

o Type

o Code

- IPv4

o Source address

o Destination Address

o Transport Layer Protocol

- IPv6

o Source address

o Destination Address

o Transport Layer Protocol and where defined by the ST author, Extension Header Type, Extension Header Fields

- TCP

o Source Port

o Destination Port

- UDP

o Source Port

o Destination Port

The evaluator shall verify that the guidance documentation indicates that each rule can identify the following actions: permit, drop, and log.

The evaluator shall verify that the guidance documentation explains how rules are associated with distinct network interfaces.

The “Traffic Flow Overview” section of the [AdminGuide] states that the security appliance supports the following protocols: ICMP, ICMPv6, IPv4, IPv6, TCP and UDP. For TCP and UDP traffic, service policies operate on traffic flows, and not just individual packets. All traffic that goes through the ASA is inspected using the Adaptive Security Algorithm and either allowed through or dropped. The “Default Traffic Flow (without ACLs)” section of the [AdminGuide] also confirms that Access Lists are required to be set up to enable traffic to flow through the security appliance. Specific permit or deny rules are required to be applied to a protocol, a source and destination IP address or Network and optionally, the source and destination ports.

The “Configure Extended ACLs” section of the [AdminGuide] describes the ‘access-list’ command and how it is used to specify the actions of deny, permit or log for a rule. It also describes how to specify the protocol or ICMP type and code.

The “Overview of Traffic to be Dropped, and the Related Syslog Messages” section of the [AdminGuide] includes examples of how the TOE should be configured to ensure that traffic not allowed in the TOE is dropped. This includes an indication that all of the required attributes of these protocols can be configured including source and destination ports/addresses, transport layer protocol and for ICMP, type and code.

The “Mandatory Traffic Flow Controls” section of the [AdminGuide] states that in its Common Criteria certified configuration, the ASA must drop certain traffic at all enabled interfaces at all times. Some of the traffic that must be dropped will be dropped at all times, regardless of configuration, other traffic will be dropped by ACLs, and still other traffic will be dropped by the “ip audit” feature. The ‘ip audit’ feature and command usage is described in this section including how to apply the “ip audit” policies to interfaces.

The “Interface Types” section of the [AdminGuide] states that the ASA interfaces which are capable of enforcing traffic filtering (via the access-group command), or terminating tunnels (e.g. via the crypto-map command) are interfaces that have been “named” (via the nameif command). The interface name is used in all configuration commands on the ASA instead of the interface type and ID (such as gigabitethernet0/1), and is therefore required before traffic can pass through the interface. For subinterfaces, a VLAN must be assigned to the subinterface (via the vlan command) before the subinterface can be named.

The “Create an Access-List and Assigning to Crypto Map” section of the [AdminGuide] describes how to create an access-list to define the traffic to be encrypted/decrypted, and create a crypto map that references that access-list, and defines the rest of the IPsec SA parameters. For evaluating IPsec traffic, the “crypto map map-name interface interface-name” command is used.

Testing Assurance Activities: Test 1: The evaluator shall use the instructions in the guidance documentation to test that stateful packet filter firewall rules can be created that permit, drop, and log packets for each of the following attributes:

- ICMPv4

o Type

o Code

- ICMPv6

o Type

o Code

- IPv4

o Source address

o Destination Address

o Transport Layer Protocol

- IPv6

o Source address

o Destination Address

o Transport Layer Protocol and where defined by the ST author, Extension Header Type, Extension Header Fields

- TCP

o Source Port

o Destination Port

- UDP

o Source Port

o Destination Port

Test 2: Repeat the test evaluation activity above to ensure that stateful traffic filtering rules can be defined for each distinct network interface type supported by the TOE.

Note that these test activities should be performed in conjunction with those of FFW_RUL_EXT.1.9 where the effectiveness of the rules is tested. The test activities for FFW_RUL_EXT.1.9 define the protocol/attribute combinations required to be tested. If those combinations are configured manually, that will fulfil the objective of these test activities, but if those combinations are configured otherwise (e.g., using automation), these test

activities may be necessary in order to ensure the guidance is correct and the full range of configurations can be achieved by a TOE administrator.

Test 1: The evaluator configured firewall rules for testing of the other FFW_RUL_EXT.1 tests (including FFW_RUL_EXT.1.9) using instructions provided within the administrative guidance and found all necessary instructions were provided accurately.

Test 2: The evaluator configured firewall rules for testing of the other FFW_RUL_EXT.1 tests (including FFW_RUL_EXT.1.9) using instructions provided within the administrative guidance and found all necessary instructions were provided accurately.

2.4.1.3 STFFW14E:FFW_RUL_EXT.1.3

TSS Assurance Activities: See FFW_RUL_EXT.1.2

See FFW_RUL_EXT.1.2.

Guidance Assurance Activities: See FFW_RUL_EXT.1.2

See FFW_RUL_EXT.1.2.

Testing Assurance Activities: See FFW_RUL_EXT.1.2

See FFW_RUL_EXT.1.2.

2.4.1.4 STFFW14E:FFW_RUL_EXT.1.4

TSS Assurance Activities: See FFW_RUL_EXT.1.2

See FFW_RUL_EXT.1.2.

Guidance Assurance Activities: See FFW_RUL_EXT.1.2

See FFW_RUL_EXT.1.2.

Testing Assurance Activities: See FFW_RUL_EXT.1.2

See FFW_RUL_EXT.1.2.

2.4.1.5 STFFW14E:FFW_RUL_EXT.1.5

TSS Assurance Activities: The evaluator shall verify that the TSS identifies the protocols that support stateful session handling. The TSS shall identify TCP, UDP, and, if selected by the ST author, also ICMP.

The evaluator shall verify that the TSS describes how stateful sessions are established (including handshake processing) and maintained.

The evaluator shall verify that for TCP, the TSS identifies and describes the use of the following attributes in session determination: source and destination addresses, source and destination ports, sequence number, and individual flags.

The evaluator shall verify that for UDP, the TSS identifies and describes the following attributes in session determination: source and destination addresses, source and destination ports.

The evaluator shall verify that for ICMP (if selected), the TSS identifies and describes the following attributes in session determination: source and destination addresses, other attributes chosen in FFW_RUL_EXT.1.5.

The evaluator shall verify that the TSS describes how established stateful sessions are removed. The TSS shall describe how connections are removed for each protocol based on normal completion and/or timeout conditions. The TSS shall also indicate when session removal becomes effective (e.g., before the next packet that might match the session is processed).

Section 6.1, Table 19 of [ST] provides the following:

FFW_RUL_EXT.1.1[FW] states that the TOE provides stateful traffic filtering of IPv4 and IPv6 network traffic. Administratively-defined traffic filter rules (access-lists) can be applied to any interface to filter traffic based on IP parameters including source and destination address, transport layer protocol, type and code, TCP and UDP port numbers. The TOE allows establishment of communications between remote endpoints, and tracks the state of each session (e.g. initiating, established, and tear-down), and will clear established sessions after proper tear-down is completed as defined by each protocol, or when session timeouts are reached.

To track the statefulness of sessions to/from and through the firewall, the TOE maintains a table of connections in various connection states and connection flags. The TOE updates the table (adding, and removing connections, and modifying states as appropriate) based on normal connection completion, configurable connection timeout limits, and by inspecting fields within the packet headers.

FFW_RUL_EXT.1.2[FW] states that the TOE supports filtering of the following protocols and enforces proper session establishment, management, and termination as defined in each protocol's RFC including proper use of:

- Addresses, type of service, fragmentation data, size and padding, and IP options including loose source routing, strict source routing, and record route as defined in RFC 791 (IPv4), and RFC 2460 (IPv6);
- Source and destination addresses, Port numbers, sequence and acknowledgement numbers, size and padding, and control bits such as SYN, ACK, FIN, and RST as defined in RFC 793 (TCP);
- Source and destination addresses, Port numbers, and length as defined in RFC 768 (UDP); and
- Session identifiers, sequence numbers, types, and codes as defined in RFC 792 (ICMPv4), and RFC 4443 (ICMPv6).

FFW_RUL_EXT.1.5[FW] states that all traffic that goes through the TOE is inspected using the Adaptive Security Algorithm and either is allowed through or dropped. If it is a new connection, the TOE has to check the packet against access control lists and perform other tasks to determine if the packet is allowed or denied. If the

connection is already established, the TOE does not need to re-check packets against the ACL; matching packets can go through the "fast" path based on attributes identified in FFW_RUL_EXT.1.5.

Guidance Assurance Activities: The evaluator shall verify that the guidance documentation describes stateful session behaviours. For example, a TOE might not log packets that are permitted as part of an existing session.

The “Stateful Inspection Overview” section of the [AdminGuide] states that all traffic that goes through the ASA is inspected using the Adaptive Security Algorithm and either allowed through or dropped.

The ASA takes into consideration whether or not this is a new connection. If it is a new connection, the ASA has to check the packet against access lists and perform other tasks to determine if the packet is allowed or denied. To perform this check, the first packet of the session goes through the "session management path," and depending on the type of traffic, it might also pass through the "control plane path."

The session management path is responsible for the following tasks:

- Performing the access list checks
- Performing route lookups
- Allocating NAT translations (xlates)
- Establishing sessions in the "fast path"

The ASA creates forward and reverse flows in the fast path for TCP traffic; the ASA also creates connection state information for connectionless protocols like UDP, so that they can also use the fast path.

If the connection is already established, the ASA does not need to re-check packets; most matching packets can go through the "fast" path in both directions. The fast path is responsible for the following tasks:

- IP checksum verification
- Session lookup
- TCP sequence number check
- NAT translations based on existing sessions
- Layer 3 and Layer 4 header adjustments

Testing Assurance Activities: Test 1: The evaluator shall configure the TOE to permit and log TCP traffic. The evaluator shall initiate a TCP session. While the TCP session is being established, the evaluator shall introduce session establishment packets with incorrect flags to determine that the altered traffic is not accepted as part of the session (i.e., a log event is generated to show the ruleset was applied). After a TCP session is successfully established, the evaluator shall alter each of the session determining attributes (source and destination addresses, source and destination ports, sequence number, flags) one at a time in order to verify that the altered packets are not accepted as part of the established session.

Test 2: The evaluator shall terminate the TCP session established per Test 1 as described in the TSS. The evaluator shall then immediately send a packet matching the former session definition in order to ensure it is not forwarded through the TOE without being subject to the ruleset.

Test 3: The evaluator shall expire (i.e., reach timeout) the TCP session established per Test 1 as described in the TSS. The evaluator shall then send a packet matching the former session in order to ensure it is not forwarded through the TOE without being subject to the ruleset.

Test 4: The evaluator shall configure the TOE to permit and log UDP traffic. The evaluator shall establish a UDP session. Once a UDP session is established, the evaluator shall alter each of the session determining attributes (source and destination addresses, source and destination ports) one at a time in order to verify that the altered packets are not accepted as part of the established session.

Test 5: The evaluator shall expire (i.e., reach timeout) the UDP session established per Test 4 as described in the TSS. The evaluator shall then send a packet matching the former session in order to ensure it is not forwarded through the TOE without being subject to the ruleset.

Test 6: If ICMP is selected, the evaluator shall configure the TOE to permit and log ICMP traffic. The evaluator shall establish a session for ICMP as defined in the TSS. Once an ICMP session is established, the evaluator shall alter each of the session determining attributes (source and destination addresses, other attributes chosen in FFW_RUL_EXT.1.5) one at a time in order to verify that the altered packets are not accepted as part of the established session.

Test 7: If applicable, the evaluator shall terminate the ICMP session established per Test 6 as described in the TSS. The evaluator shall then immediately send a packet matching the former session definition in order to ensure it is not forwarded through the TOE without being subject to the ruleset.

Test 8: The evaluator shall expire (i.e., reach timeout) the ICMP session established per Test 6 as described in the TSS. The evaluator shall then send a packet matching the former session in order to ensure it is not forwarded through the TOE without being subject to the ruleset.

The evaluation team performed the tests specified in the assurance activity and confirmed that the traffic filter firewall rules operate as expected in each test scenario. The tests were performed using both IPv4 and IPv6.

Test 1: The TOE was configured to log the occurrence of dropped out-of-state packets. The evaluator utilized a script on the GSS Test server to establish TCP connections with an FTP server by passing traffic through the TOE. The script tests the TOE's treatment of ancillary packets received before, during, and after session establishment. A TCP session is established by a sequence of packets exchanged between two peers. This test injects packets with invalid flags at various points in this exchange. Packet captures representing the packets sent into the TOE (IN pcap) and the packets detected coming out of the TOE (OUT pcap) were generated. The evaluator concluded from the difference between the two captures that the TOE correctly detected the out of state packets and dropped them.

Test 2: Continuing from Test 1, the evaluator configured the TOE with UDP session and TCP session timeouts of 30 seconds and 5 minutes, respectively (the minimums allowed by the TOE). Once configured, the evaluator ran the GSS Test server test script to execute a test similar to the previous test. The test takes the following steps.

1. Unlike the prior test, this test does not send any out of state packets before establishing a TCP session with the FTP server,

2. After fully establishing the TCP session (with the GSS FTP server), the script gracefully terminates the TCP session and then waits/delays 11 seconds (to allow for TOEs that enforce an additional TCP end timeout) so that the TOE can fully close the TCP session and

3. The script then attempts to send two final TCP packets belonging to the now closed FTP control session to test whether or not the TOE correctly blocks them.

Packet captures representing the packets sent into the TOE (IN pcap) and the packets detected coming out of the TOE (OUT pcap) were generated. The evaluator concluded from the difference between the two captures that the TOE correctly detected the out of state packets and dropped them. The evaluator also observed that no packets sent after the TCP session terminated were processed as a part of the original TCP session.

Test 3: Repeated Test 2, expiring the session rather than explicitly terminating the session. The evaluator observed that no packets sent after a session expiration were treated by the TOE as part of the original TCP session.

Test 4: Using guidance, the evaluator configured the TOE to allow and log network traffic for a valid UDP session. The evaluator started transmitting packets from a test server to establish a valid UDP session. The packets being sent were constructed such that they would be permitted to pass through the TOE. The evaluator also sent packets with incorrect source and destination addresses and incorrect source and destination ports. The evaluator confirmed (through logs and packet captures) that all nonmatching packets are not accepted as part of the established session.

Test 5: The evaluator performed test 3 (using an expired session) using a UDP session rather than a TCP session. The evaluator observed that no packets sent after a session expiration were treated by the TOE as a part of the original UDP session.

Test 6: Not applicable. ICMP was not selected.

Test 7: Not applicable. ICMP was not selected.

Test 8: Not applicable. ICMP was not selected.

2.4.1.6 STFFW14E:FFW_RUL_EXT.1.6

TSS Assurance Activities: The evaluator shall verify that the TSS identifies the following as packets that will be automatically dropped and are counted or logged:

a) Packets which are invalid fragments, including a description of what constitutes an invalid fragment

b) Fragments that cannot be completely re-assembled

c) Packets where the source address is defined as being on a broadcast network

d) Packets where the source address is defined as being on a multicast network

e) Packets where the source address is defined as being a loopback address

f) The TSF shall reject and be capable of logging network packets where the source or destination address of the network packet is defined as being unspecified (i.e. 0.0.0.0) or an address 'reserved for future use' (i.e. 240.0.0.0/4) as specified in RFC 5735 for IPv4;

g) The TSF shall reject and be capable of logging network packets where the source or destination address of the network packet is defined as an 'unspecified address' or an address 'reserved for future definition and use' (i.e. unicast addresses not in this address range: 2000::/3) as specified in RFC 3513 for IPv6;

h) Packets with the IP options: Loose Source Routing, Strict Source Routing, or Record Route specified

i) Other packets defined in FFW_RUL_EXT.1.6 (if any).

Section 6.1, Table 19 (FFW_RUL_EXT.1.6[FW]) of [ST] states that the TOE can be configured to implement default denial of various mal-formed packets/fragments, and other illegitimate network traffic, and can be configured to log the packets/frames that were dropped.

This section specifies all the traffic that will be denied and logged consistent with the traffic specified in FFW_RUL_EXT.1.6 and FFW_RUL_EXT.1.7. This includes specification of packets which are invalid fragments and a description of what constitutes an invalid fragment. The evaluator verified that all traffic listed in this assurance activity is identified using the following mapping of assurance activity (a-i) and TSS list (#1-12).

a) invalid fragments (item 1)
b) fragments (item 2)
c) source address is broadcast (item 5)
d) source address is multicast (item 6)
e) source address is loopback (item 7)
f) unspecified or reserved IPv4 (item 9)
g) unspecified or reserved IPv6 (item 10
h) Packets with the IP options (item 11)
i) other (item 12)

Guidance Assurance Activities: The evaluator shall verify that the guidance documentation describes packets that are discarded and potentially logged by default. If applicable protocols are identified, their descriptions need to be consistent with the TSS. If logging is configurable, the evaluator shall verify that applicable instructions are provided to configure auditing of automatically rejected packets.

The “Overview of Traffic to be Dropped, and the Related Syslog Messages” section in the [AdminGuide] describes how packets are discarded and potentially logged by default. It also provides instructions in this section and in the “Logging and Log Messages” section for configuring audit of automatically rejected packets.

Testing Assurance Activities: Both IPv4 and IPv6 shall be tested for items a), b), c), d), and e) of the SFR element. Both IPv4 and IPv6 shall be tested for item i) unless the rule definition is specific to IPv4 or IPv6. Note: f), g), and h) are specific to IPv4 or IPv6 and shall be tested accordingly.

Test 1: The evaluator shall test each of the conditions for automatic packet rejection in turn. In each case, the TOE should be configured to allow all network traffic and the evaluator shall generate a packet or packet fragment that is to be rejected. The evaluator shall use packet captures to ensure that the unallowable packet or packet fragment is not passed through the TOE.

Test 2: For each of the cases above, the evaluator shall use any applicable guidance to enable dropped packet logging or counting. In each case above, the evaluator shall ensure that the rejected packet or packet fragment was recorded (either logged or an appropriate counter incremented).

Test 1: The evaluator sent a series of packets to the TOE, while capturing both the packets entering into the TOE (Test Network/IN pcap) as well as those exiting the TOE (Probe Network/OUT pcap). The evaluator configured firewall rules to allow network traffic and enabled rejected packet logging. The evaluator generated the following types of packets which the TOE rejected and logged.

- Invalid Fragment IPv4 and IPv6
- Incomplete Fragment IPv4 and IPv6
- Invalid Broadcast Source Address IPv4
- Invalid Multicast Source Address IPv4 and IPv6
- Invalid Loopback Source Address IPv4 and IPv6
- Invalid Future Source and Destination Address IPv4 and IPv6
- Invalid Loose and Strict Source Routing and Record Route IPv4

In addition to the PP required default rules, the security target identifies several additional rules which were also tested.

1. The evaluator ensured that the TOE rejected and was capable of logging IPv4 packets with destination IP address equal to 0.0.0.0.
2. The evaluator ensured that the TOE rejected and was capable of logging IPv4 packets with source IP address equal to 0.0.0.0.
3. The evaluator ensured that the TOE rejected and was capable of logging packets with the first octet of the source IP address equal to zero.
4. The evaluator ensured that the TOE rejected and was capable of logging packets with the network part of the source IP address equal to all 0’s.
5. The evaluator ensured that the TOE rejected and was capable of logging packets with the network part of the source IP address equal to all 1’s.
6. The evaluator ensured that the TOE rejected and was capable of logging packets with the host part of the source IP address equal to all 1’s.
7. The evaluator ensured that the TOE rejected and was capable of logging packets with the host part of the source IP address equal to all 0’s.

8. The evaluator ensured that the TOE rejected and was capable of logging ICMP error packets when the ICMP error messages are not related to any session already established in the TOE (also tested for ICMPv6).
9. The evaluator ensured that the TOE rejected and was capable of logging network packets when the appliance is not able to find any established connection related to the frame embedded in the ICMPv6 error message.
10. The evaluator ensured that the TOE rejected and was capable of logging network packets when an ICMP echo request/reply packet was received with a malformed code(non-zero).

Test 2: The evaluator observed that the TOE correctly blocked each of the packets directed to it and confirmed that the TOE correctly logged all rejected/dropped/blocked packets.

2.4.1.7 STFFW14E:FFW_RUL_EXT.1.7

TSS Assurance Activities: The evaluator shall verify that the TSS explains how the following traffic can be dropped and counted or logged:

a) Packets where the source address is equal to the address of the network interface where the network packet was received

b) Packets where the source or destination address of the network packet is a link-local address

c) Packets where the source address does not belong to the networks associated with the network interface where the network packet was received, including a description of how the TOE determines whether a source address belongs to a network associated with a given network interface

Section 6.1, Table 19 (FFW_RUL_EXT.1.7[FW]) of [ST] states that the TOE can be configured to implement default denial of various mal-formed packets/fragments, and other illegitimate network traffic, and can be configured to log the packets/frames that were dropped. The TOE can be configured to deny and log traffic by defining policies with the “ip audit name” command, specifying the “drop” action, and applying the policy or policies to each enabled interface. Each signature has been classified as either “informational”, or “attack”. Using the “info” and “attack” keywords in the “ip audit name” command defines the action the TOE will take for each signature classification. This section further specifies all the traffic that will be denied and logged consistent with the traffic specified in FFW_RUL_EXT.1.6[FW] and FFW_RUL_EXT.1.7[FW]. The evaluator verified that all traffic listed in this assurance activity is identified.

The TOE determines whether or not a network packet’s source address belongs to the networks associated with the network interface where the network packet was received and will deny the packet if the IP address has been spoofed, if the route for the packet does not match the interface on which it arrived (the TOE uses the ‘ip verify reverse-path command’ to determine this), or if the packet matches a connection, but arrives on a different interface from the interface on which the connection began.

Guidance Assurance Activities: The evaluator shall verify that the guidance documentation describes how the TOE can be configured to implement the required rules. If logging is configurable, the evaluator shall verify that applicable instructions are provided to configure auditing of automatically rejected packets.

The "Overview of Traffic to be Dropped and the Related Syslog Messages" section in the [AdminGuide] describes how packets are discarded and potentially logged by default. It also provides instructions in this section and in the "Logging and Log Messages" section for configuring audit of automatically rejected packets.

Testing Assurance Activities: Test 1: The evaluator shall configure the TOE to drop and log network traffic where the source address of the packet matches that of the TOE network interface upon which the traffic was received. The evaluator shall generate suitable network traffic to match the configured rule and verify that the traffic is dropped and a log message generated.

Test 2: The evaluator shall configure the TOE to drop and log network traffic where the source IP address of the packet fails to match the network reachability information of the interface to which it is targeted, e.g. if the TOE believes that network 192.168.1.0/24 is reachable through interface 2, network traffic with a source address from the 192.168.1.0/24 network should be generated and sent to an interface other than interface 2. The evaluator shall verify that the network traffic is dropped and a log message generated.

The tests were performed using both IPv4 and IPv6.

Test 1: The evaluator configured the TOE to drop and log network traffic where the source address of the packet matches that of the TOE network interface upon which the traffic was received. The evaluator generated suitable traffic to match the configured rule, while capturing both the packets entering into the TOE (Test Network/IN pcap) as well as those exiting the TOE (Probe Network/OUT pcap). The evaluator confirmed via packet capture and logs that the traffic is dropped and a log message is generated.

Test 2: The evaluator configured the TOE to drop and log network traffic where the source IP address of the packet fails to match the network reachability information of the interface to which it is targeted. The evaluator generated suitable traffic to match the configured rule, while capturing both the packets entering into the TOE (Test Network/IN pcap) as well as those exiting the TOE (Probe Network/OUT pcap). The evaluator confirmed via packet capture and logs that the traffic is dropped and a log message is generated.

2.4.1.8 STFFW14E:FFW_RUL_EXT.1.8

TSS Assurance Activities: The evaluator shall verify that the TSS describes the algorithm applied to incoming packets, including the processing of default rules, determination of whether a packet is part of an established session, and application of administrator defined and ordered ruleset.

Section 6.1, Table 19 of [ST] provides the following:

FFW_RUL_EXT.1.5[FW] states that all traffic that goes through the TOE is inspected using the Adaptive Security Algorithm and either is allowed through or dropped.

If it is a new connection, the TOE has to check the packet against access control lists and perform other tasks to determine if the packet is allowed or denied. To perform this check, the first packet of the session goes through the "session management path," and depending on the type of traffic, it might also pass through the "control plane path."

The session management path is responsible for the following tasks:

- Performing the access list checks
- Performing route lookups
- Allocating NAT translations (xlates)
- Establishing sessions in the "fast path"

The TOE creates forward and reverse flows in the fast path for TCP traffic; the TOE also creates connection state information for connectionless protocols like UDP, ICMP (when you enable ICMP inspection), so that they can also use the fast path.

If the connection is already established, the TOE does not need to re-check packets against the ACL; matching packets can go through the "fast" path based on attributes identified in FFW_RUL_EXT.1.5[FW]. The fast path is responsible for the following tasks:

- IP checksum verification
- Session lookup
- TCP sequence number check
- NAT translations based on existing sessions
- Layer 3 and Layer 4 header adjustments

FFW_RUL_EXT.1.6[FW] states that the TOE can be configured to implement default denial of various mal-formed packets/fragments, and other illegitimate network traffic, and can be configured to log the packets/frames that were dropped.

FFW_RUL_EXT.1.8[FW] states that TOE administrators have control over the sequencing of access control entries (ACEs) within an access control list (ACL) to be able to set the sequence in which ACEs are applied within any ACL. The entries within an ACL are always applied in a top-down sequence, and the first entry that matches the traffic is the one that's applied, regardless of whether there may be a more precise match for the traffic further down in the ACL. By changing the ordering/numbering of entries within an ACL, the administrator changes the sequence in which the entries are compared to network traffic flows.

Guidance Assurance Activities: The evaluator shall verify that the guidance documentation describes how the order of stateful traffic filtering rules is determined and provides the necessary instructions so that an administrator can configure the order of rule processing.

The "Access Lists" section of the [AdminGuide] states that the **access-list** command operates on a first-match basis. Therefore, the last rule added to the access list is the last rule checked. Administrators must take note of this when entering the initial rules during the configuration, as it may impact the remainder of the rule parsing.

Testing Assurance Activities: Test1: If the TOE implements a mechanism that ensures that no conflicting rules can be configured, the evaluator shall try to configure two conflicting rules and verify that the TOE rejects the conflicting rule(s). It is important to verify that the mechanism is implemented in the TOE but not in the non-TOE environment. If the TOE does not implement a mechanism that ensures that no conflicting rules can be configured, the evaluator shall devise two equal stateful traffic filtering rules with alternate operations - permit and drop. The rules should then be deployed in two distinct orders and in each case the evaluator shall ensure that the first rule is enforced in both cases by generating applicable packets and using packet capture and logs for confirmation. (TD0545 applied)

Test 2: The evaluator shall repeat the procedure above, except that the two rules should be devised where one is a subset of the other (e.g., a specific address vs. a network segment). Again, the evaluator should test both orders to ensure that the first is enforced regardless of the specificity of the rule.

These tests were performed using both IPv4 and IPv6.

Test 1: The evaluator attempted to configure the TOE (according to the admin guide) with two firewall rules using the same matching criteria, where one rule would permit while the other deny traffic. The evaluator configured a permit rule first with the second rule (deny). The evaluator confirmed that the traffic was permitted. Subsequently, the evaluator configured a deny rule first with the second rule (permit). The evaluator observed that the traffic was blocked.

Test 2: Continuing test 1, the evaluator repeated the procedure above, except the evaluator changed the rules to make one a subset of the other, and then tested both orders. The evaluator confirmed that the first is enforced regardless of the specificity of the rule.

2.4.1.9 STFFW14E:FFW_RUL_EXT.1.9

TSS Assurance Activities: The evaluator shall verify that the TSS describes the process for applying stateful traffic filtering rules and also that the behavior (either by default, or as configured by the administrator) is to deny packets when there is no rule match unless another required conditions allows the network traffic (i.e., FFW_RUL_EXT.1.5 or FFW_RUL_EXT.2.1).

Section 6.1, Table 19 of [ST] provides the following:

FFW_RUL_EXT.1.8[FW] states that the entries within an ACL are always applied in a top-down sequence, and the first entry that matches the traffic is the one that's applied, regardless of whether there may be a more precise match for the traffic further down in the ACL. By changing the ordering/numbering of entries within an ACL, the administrator changes the sequence in which the entries are compared to network traffic flows.

FFW_RUL_EXT.1.9[FW] states that an implicit "deny-all" rule is applied to all interfaces to which any traffic filtering rule has been applied. The implicit deny-all rule is executed after all admin-defined rules have been executed, and will result in dropping all traffic that has not been explicitly permitted, or explicitly denied.

Guidance Assurance Activities: The evaluator shall verify that the guidance documentation describes the behavior if no rules or special conditions apply to the network traffic. If the behavior is configurable, the evaluator shall verify that the guidance documentation provides the appropriate instructions to configure the behavior to deny packets with no matching rules.

The "Default Traffic Flow (without ACLs)" section of the [AdminGuide] states that the ASA, by default, is configured with a default DHCP address pool. The outbound interface disallows all external to internal data flows. The administrator needs to be aware of this, and ensure that the correct policy for the organization is installed and committed before users are permitted to use the security appliance. Access Lists are required to be set up to enable traffic to flow through the security appliance. Specific permit or deny rules are required to be applied to a protocol, a source and destination IP address or Network and optionally, the source and destination ports.

Table 5 and 6 provide a list of the default/implicit traffic flow policy applied between the "outside" and "inside" networks when no ACLs have been applied to outside or inside interfaces of the ASA.

Testing Assurance Activities: For each attribute in FFW_RUL_EXT.1.2, the evaluator shall construct a test to demonstrate that the TOE can correctly compare the attribute from the packet header to the ruleset, and shall demonstrate both the permit and deny for each case. It shall also be verified that a packet is dropped if no matching rule can be identified for the packet. The evaluator shall check the log in each case to confirm that the relevant rule was applied. The evaluator shall record a packet capture for each test to demonstrate the correct TOE behaviour.

The evaluator defined several tests variations to exercise the attributes and rules from FFW_RUL_EXT.1.2. The evaluator generated traffic to match specific aspects of the configured firewall rule set and confirmed that all attributes demonstrated permit, deny and log for each test case. Some examples of test variations exercised rules for ICMP, IPv4, IPv6, TCP, and UDP traffic demonstrating both permit and deny rules for these protocols.

2.4.1.10 STFFW14E:FFW_RUL_EXT.1.10

TSS Assurance Activities: The evaluator shall verify that the TSS describes how the TOE tracks and maintains information relating to the number of half-open TCP connections. The TSS should identify how the TOE behaves when the administratively defined limit is reached and should describe under what circumstances stale half-open connections are removed (e.g. after a timer expires).

Section 6.1, Table 19 (FFW_RUL_EXT.1.10[FW]) of [ST] states that TOE administrators can configure the maximum number of half-open TCP connections allowed using the "set connection embryonic-conn-max 0-65535" in the service-policy command. After the configured limit is reached, the TOE will act as a proxy for the server and generates a SYN-ACK response to new client SYN requests. When the ASA receives an ACK back from the client, it can then authenticate that the client is real and allow the connection to the server. If an ACK is not received in the configurable time frame, the session is closed, resource is returned to the free pool, and it will be counted. The default idle time until a TCP half-open connection closes is 10 minutes.

Guidance Assurance Activities: The evaluator shall verify that the guidance documentation describes the behaviour of imposing TCP half-open connection limits and its default state if unconfigured. The evaluator shall verify that the guidance clearly indicates the conditions under which new connections will be dropped e.g. perdestination or per-client.

The “Protect from SYN Flood DoS Attack (TCP Intercept)” section of the [AdminGuide] describes how the administrator can limit the number of embryonic connections to help prevent SYN flooding attacks. An embryonic connection is a connection request that has not finished the necessary handshake between source and destination.

When the embryonic connection threshold of a connection is crossed, the ASA acts as a proxy for the server and generates a SYN-ACK response to the client SYN request using the SYN cookie method (see Wikipedia for details on SYN cookies). When the ASA receives an ACK back from the client, it can then authenticate that the client is real and allow the connection to the server. The component that performs the proxy is called TCP Intercept.

The end-to-end process for protecting a server from a SYN flood attack involves setting connection limits, enabling TCP Intercept statistics, and then monitoring the results.

This section further provides the commands in step 3 to set the embryonic connection limits as follows:

- **set connection embryonic-conn-max** n-The maximum number of simultaneous embryonic connections allowed, between 0 and 2000000. The default is 0, which allows unlimited connections.

- **set connection per-client-embryonic-max** n-The maximum number of simultaneous embryonic connections allowed per client, between 0 and 2000000. The default is 0, which allows unlimited connections.

Testing Assurance Activities: Test 1: The evaluator shall define a TCP half-open connection limit on the TOE. The evaluator shall generate TCP SYN requests to pass through the TOE to the target system using a randomised source IP address and common destination IP address. The number of SYN requests should exceed the TCP half-open threshold defined on the TOE. TCP SYN-ACK messages should not be acknowledged. The evaluator shall verify through packet capture that once the defined TCP half-open threshold has been reached, subsequent TCP SYN packets are not transmitted to the target system. The evaluator shall verify that when the configured threshold is reached that, depending upon the selection, either a log entry is generated or a counter is incremented.

The evaluator configured a TCP half-open connection limit on the TOE. The evaluator then generated TCP SYN requests which would pass through the TOE to a target system using a randomized source IP address and common destination IP address. The number of SYN requests sent exceeded the TCP half-open threshold defined on the TOE. TCP SYN-ACK messages were not acknowledged. Using a packet capture, the evaluator verified that once the defined TCP half-open threshold had been reached, subsequent TCP SYN packets are not transmitted to the target system. The evaluator verified that when the configured threshold is reached the TOE behaved as claimed in the Security Target.

Component TSS Assurance Activities: The evaluator shall verify that the TSS provides a description of the TOE's initialization/startup process, which clearly indicates where processing of network packets begins to take place, and provides a discussion that supports the assertion that packets cannot flow during this process.

The evaluator shall verify that the TSS also include a narrative that identifies the components (e.g., active entity such as a process or task) involved in processing the network packets and describe the safeguards that would prevent packets flowing through the TOE without applying the ruleset in the event of a component failure. This could include the failure of a component, such as a process being terminated, or a failure within a component, such as memory buffers full and cannot process packets. The description shall also include a description how the TOE behaves in the situation where the traffic exceeds the amount of traffic the TOE can handle and how it is ensured that also in this condition stateful traffic filtering rules are still applied so that traffic does not pass that shouldn't pass according to the specified rules.

Section 6.1, Table 19 (FFW_RUL_EXT.1.1[FW]) in [ST] states that during initialization/startup (while the TOE is booting) the configuration has yet to be loaded, and no traffic can flow through any of its interfaces. No traffic can flow through the TOE interfaces until the POST has completed, and the configuration has been loaded. If any aspect of the POST fails during boot, the TOE will reload without forwarding traffic. If a critical subcomponent of the TOE, such as the clock or cryptographic modules, fails while the TOE is in an operational state, the TOE will reload, which stops the flow of traffic. If a subcomponent such as a network interface, which is not critical to the operation of the TOE, but may be critical to one or more traffic flows, fails while the TOE is operational, the TOE will continue to function, though all traffic flows through the failed network interface(s) will be dropped.

When traffic exceeds the maximum rate the TOE can handle, the TOE drops the excess traffic and ensures that no traffic that wouldn't pass stateful traffic filtering rules would be passed through.

Component Guidance Assurance Activities: The guidance documentation associated with this requirement is assessed in the subsequent test evaluation activities.

Refer to the test evaluation activities for FFW_RUL_EXT.1 elements below.

Component Testing Assurance Activities: Test 1: The evaluator shall attempt to get network traffic to flow through the TOE while the TOE is being initialized. A steady flow of network packets that would otherwise be denied by the ruleset should be sourced and be directed at a host. The evaluator shall verify using a packet sniffer that none of the generated network traffic is permitted through the firewall during initialization.

Test 2: The evaluator shall attempt to get network traffic to flow through the TOE while the TOE is being initialized. A steady flow of network packets that would be permitted by the ruleset should be sourced and be directed at a host. The evaluator shall verify using a packet sniffer that none of the generated network traffic is permitted through the firewall during initialization and is only permitted once initialization is complete.

Note: The remaining testing associated with application of the ruleset is addressed in the subsequent test evaluation activities.

Test 1: The evaluator configured the TOE to block network traffic directed at a specific destination. The evaluator generated suitable traffic to match the configured rule, while capturing both the packets entering into the TOE (Test Network/IN pcap) as well as those exiting the TOE (Probe Network/OUT pcap). The evaluator rebooted the TOE and monitored traffic on both sides of the TOE during the reboot process. The evaluator confirmed via packet capture and logs that that while packets were being delivered to the TOE ingress side, no matching packets were being passed from the egress side.

Test 2: Using guidance, the evaluator configured the TOE to permit network traffic directed at a specific destination. The evaluator generated suitable traffic to match the configured rule, while capturing both the packets entering into the TOE (Test Network/IN pcap) as well as those exiting the TOE (Probe Network/OUT pcap). The evaluator rebooted the TOE and monitored traffic on both sides of the TOE during the reboot process. The evaluator observed that there is a gap during which no traffic is passed shortly after the reboot is started until just prior to the login prompt being presented by the TOE. This demonstrates that none of the generated network traffic is permitted through the firewall during initialization and is only permitted once initialization is complete.

2.4.2 Stateful Filtering of Dynamic Protocols (STFFW14E:FFW_RUL_EXT.2)

2.4.2.1 STFFW14E:FFW_RUL_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS identifies the protocols that can cause the automatic creation of dynamic packet filtering rules. In some cases rather than creating dynamic rules, the TOE might establish stateful sessions to support some identified protocol behaviors.

The evaluator shall verify that the TSS explains the dynamic nature of session establishment and removal. The TSS also shall explain any logging ramifications.

The evaluator shall verify that for each of the protocols selected, the TSS explains the dynamic nature of session establishment and removal specific to the protocol.

Section 6.1, Table 19 (FFW_RUL_EXT.2[FW]) of [ST] states that the TOE supports dynamic establishment of secondary network sessions (e.g., FTP). The TOE will manage establishment and teardown of the following protocol:

- FTP (File Transfer Protocol) is a TCP protocol supported in either active or passive mode:
  - In active mode the client initiates the control session, and the server initiates the data session to a client port provided by the client;
  - For active FTP to be allowed through the TOE, the firewall rules must explicitly permit the control session from the client to the server, and “inspect ftp” must be enabled. The TOE will then

explicitly permit a control session to be initiated from the client to the server, and implicitly permit data sessions to be initiated from the server to the client while the control session is active.

- In passive (PASV) mode, the client initiates the control session, and the client also initiates the data session to a secondary port provided to the client by the server.

For passive FTP to be permitted through the TOE, the firewall rules must explicitly permit the control session from the client to the server, and "inspect ftp" must be enabled with the "match passive-ftp" option enabled. That feature will cause the TOE to look for the PASV or EPSV commands in the FTP control traffic and for the server's destination port, and dynamically permit the data session.

Section 6.1, Table 19 (FFW_RUL_EXT.1.6 [FW]) of [ST] describes how the TOE can be configured to log traffic by defining policies. As described below, the [AdminGuide] provides instructions for creating an FTP inspection policy map and configuring the action to be performed on the matching traffic such as "log".

Component Guidance Assurance Activities: The evaluator shall verify that the guidance documentation describes dynamic session establishment capabilities.

The evaluator shall verify that the guidance documentation describes the logging of dynamic sessions consistent with the TSS.

The "Application Layer Protocol Inspection" section of the [AdminGuide] describes dynamic session establishment capabilities. It states that Inspection engines are required for services that embed IP addressing in formation in the user data packet or that open secondary channels on dynamically assigned ports (i.e., dynamic protocol). Many protocols open secondary TCP or UDP ports. The initial session on a well-known port is used to negotiate dynamically assigned port numbers. When application inspection is enabled for a service that uses dynamically assigned ports, the ASA monitors sessions to identify the dynamic port assignments, and permits data exchange on these ports for the duration of the specific session.

Instructions for creating an FTP inspection policy map are included in this section including how to configure the action to be performed on the matching traffic such as "log".

Component Testing Assurance Activities: Test 1: The evaluator shall define stateful traffic filtering rules to permit and log traffic for each of the supported protocols and drop and log TCP and UDP ports above 1024. Subsequently, the evaluator shall establish a connection for each of the selected protocols in order to ensure that it succeeds. The evaluator shall examine the generated logs to verify they are consistent with the guidance documentation.

Test 2: Continuing from Test 1, the evaluator shall determine (e.g., using a packet sniffer) which port above 1024 opened by the control protocol, terminate the connection session, and then verify that TCP or UDP (depending on the protocol selection) packets cannot be sent through the TOE using the same source and destination addresses and ports.

Test 3: For each additionally supported protocol, the evaluator shall repeat the procedure above for the protocol. In each case the evaluator must use the applicable RFC or standard in order to determine what range of ports to block in order to ensure the dynamic rules are created and effective.

These tests are performed using FTP and FTPv6.

Test 1: The evaluator first configured the TOE and created two firewall rules - the first rule allows FTP traffic (TCP, port 21) for a specific destination, and a second rule (to block TCP traffic for ports above 1024). The evaluator then used a test server to attempt FTP connections through the TOE to an FTP server. An FTP session was successfully established through the TOE. Packet captures were obtained for this connection. The packet captures show that the TOE did not block any of the FTP related connections, even though the TOE was configured to block TCP packets with a destination port greater than 1024 (put another way, the TOE correctly, dynamically allowed establishment of FTP data sessions).

Test 2: Continuing Test 1, after closing the FTP session, the evaluator attempted to send TCP packets through the FTP session data ports identified in Test 1. The evaluator confirmed that these packets were not sent through the TOE.

Test 3: Not applicable. There are no additional supported protocols.

2.5 Identification and Authentication (FIA)

2.5.1 Authentication Failure Management (NDcPP22e:FIA_AFL.1)

2.5.1.1 NDcPP22e:FIA_AFL.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.1.2 NDcPP22e:FIA_AFL.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it contains a description, for each supported method for remote administrative actions, of how successive unsuccessful authentication attempts are detected and tracked. The TSS shall also describe the method by which the remote administrator is prevented from successfully logging on to the TOE, and the actions necessary to restore this ability.

The evaluator shall examine the TSS to confirm that the TOE ensures that authentication failures by remote administrators cannot lead to a situation where no administrator access is available, either permanently or temporarily (e.g. by providing local logon which is not subject to blocking).

Section 6.1, Table 19 (FIA_AFL.1]) of [ST] states that the TOE provides the privileged administrator the ability to specify the maximum number of unsuccessful authentication attempts (between 1 and 16) before a privileged administrator or non-privileged administrator is locked out. When a privileged administrator or non-privileged administrator attempting to login reaches the administratively set maximum number of failed authentication attempts, the user will not be granted access to the administrative functionality of the TOE until a privileged administrator resets the user's number of failed login attempts (i.e., unlocks) through the administrative CLI (local access is permitted).

Component Guidance Assurance Activities: The evaluator shall examine the guidance documentation to ensure that instructions for configuring the number of successive unsuccessful authentication attempts and time period (if implemented) are provided, and that the process of allowing the remote administrator to once again successfully log on is described for each 'action' specified (if that option is chosen). If different actions or mechanisms are implemented depending on the secure protocol employed (e.g., TLS vs. SSH), all must be described.

The evaluator shall examine the guidance documentation to confirm that it describes, and identifies the importance of, any actions that are required in order to ensure that administrator access will always be maintained, even if remote administration is made permanently or temporarily unavailable due to blocking of accounts as a result of FIA_AFL.1.

The “Account Lockout after Failed Login Attempts” section in the [AdminGuide] describes how to configure the number of unsuccessful authentication attempts after which local accounts will be locked. To limit the number of consecutive failed local login attempts that the ASA allows any given user account (with the exception of users with a privilege level of 15; this feature does not affect level 15 users), “**aaa local authentication attempts max-fail”** command is used in global configuration mode.

The ASA can be configured to enforce that requirement on all interfaces (including the local serial console), but the guidance warns that caution should be used when applying these configurations to the local serial console interface to avoid a situation in which all local admin accounts become locked. The guide advises that in order to enforce the ability to lockout any local account after consecutive failed logins, the administrator should ensure that no privilege level 15 accounts exist in the local user database. In a configuration where no privilege level 15 accounts exist in the local user database, it’s still possible to login to any admin interface (serial, or ASDM) using an account that has privilege level 15 if that account is authenticated to a remote AAA server. In such cases, it would be expected that the remote AAA server would enforce locking of accounts after successive failed login attempts.

The administrator uses the ‘clear aaa local user lockout’ command to clear the lockout status of the specified users and set their failed-attempts counter to 0.

Component Testing Assurance Activities: The evaluator shall perform the following tests for each method by which remote administrators access the TOE (e.g. any passwords entered as part of establishing the connection protocol or the remote administrator application):

a) Test 1: The evaluator shall use the operational guidance to configure the number of successive unsuccessful authentication attempts allowed by the TOE (and, if the time period selection in FIA_AFL.1.2 is included in the ST, then the evaluator shall also use the operational guidance to configure the time period after which access is re-enabled). The evaluator shall test that once the authentication attempts limit is reached, authentication attempts with valid credentials are no longer successful.

b) Test 2: After reaching the limit for unsuccessful authentication attempts as in Test 1 above, the evaluator shall proceed as follows.

If the administrator action selection in FIA_AFL.1.2 is included in the ST then the evaluator shall confirm by testing that following the operational guidance and performing each action specified in the ST to re-enable the remote administrator's access results in successful access (when using valid credentials for that administrator).

If the time period selection in FIA_AFL.1.2 is included in the ST then the evaluator shall wait for just less than the time period configured in Test 1 and show that an authorisation attempt using valid credentials does not result in successful access. The evaluator shall then wait until just after the time period configured in Test 1 and show that an authorisation attempt using valid credentials results in successful access.

Test 1: The evaluator configured a limit on failed authentication attempts (i.e., 3 failures). The evaluator then performed more login attempts using incorrect credentials than the configured limit. The evaluator observed that the use of valid credentials immediately after exceeding the limit does not result in a successful login. The evaluator then unlocked the account (using procedures from guidance), observed that the user could login successfully with the correct password and that the count of failed login attempts was reset to zero. These steps were completed for SSH and HTTPS authentication.

Test 2: The TOE supports does not support unlocking based on time.

2.5.2 Password Management - per TD0792 (NDcPP22e:FIA_PMG_EXT.1)

2.5.2.1 NDcPP22e:FIA_PMG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check that the TSS lists the supported special character(s) for the composition of administrator passwords.

The evaluator shall check the TSS to ensure that the minimum_password_length parameter is configurable by a Security Administrator.

The evaluator shall check that the TSS lists the range of values supported for the minimum_password_length parameter. The listed range shall include the value of 15.

(TD0792 applied)

Section 6.1, Table 19 (FIA_PMG_EXT.1) in [ST] states that the TOE supports the local definition of users with corresponding passwords. The passwords can be composed of any combination of upper and lower case letters, numbers, and special characters as listed in the SFR. Minimum password length is settable by the Authorized Administrator, and supports passwords of 3 to 127 characters.

Component Guidance Assurance Activities: The evaluator shall examine the guidance documentation to determine that it:

a) identifies the characters that may be used in passwords and provides guidance to security administrators on the composition of strong passwords, and

b) provides instructions on setting the minimum password length and describes the valid minimum password lengths supported.

The “Passwords” section of the [AdminGuide] provides the following guidance for the certified configuration:

In the certified configuration, password length should be settable by the administrator and can support 15 characters long, and have some complexity requirements enforced.

The following is a list of characters that can be used within passwords:

- 26 Upper case letters (A - Z)
- 26 Lower case letter (a - z)
- 10 Numbers (0 - 9)
- !"#$%&'()*+,-./:;<@[\`{|=>?]^_}~

These are a total of 93 characters that may be used to construct a password. The use of the space character is prohibited.

Administrators must ensure that when creating or changing a password, the following requirements are met:

1. Passwords must:

   - be settable and can support 15 characters long (NOTE: no lower than 8 minimum)
   - include mixed-case alphabetic characters
   - include at least 1 numeric character
   - include at least 1 special character

2. Passwords must not include:

- birthdays
- names (parents, family, spouse, pets, favorite sports player)
- sports teams
- towns, cities or countries

The "Password Policies" section of the [AdminGuide] describes how to configure the minimum password length using the `password-policy minimum-length <3-127>` command. The recommendation is for the password length to be set to 8 or greater.

Component Testing Assurance Activities: The evaluator shall perform the following tests.

Test 1: The evaluator shall compose passwords that meet the requirements in some way. For each password, the evaluator shall verify that the TOE supports the password. While the evaluator is not required (nor is it feasible) to test all possible compositions of passwords, the evaluator shall ensure that all characters, and a minimum length listed in the requirement are supported and justify the subset of those characters chosen for testing.

Test 2: The evaluator shall compose passwords that do not meet the requirements in some way. For each password, the evaluator shall verify that the TOE does not support the password. While the evaluator is not required (nor is it feasible) to test all possible compositions of passwords, the evaluator shall ensure that the TOE enforces the allowed characters and the minimum length listed in the requirement and justify the subset of those characters chosen for testing.

Test 1 & 2 - The evaluator attempted to set/change a password for a user's account using several attempts. Throughout those attempts, every upper case letter, lower case letter, digit, and special character (as specified by the SFR in [ST]) were used in a password. The evaluator confirmed that a minimum length of 8 was required by attempting to set passwords with 7 characters (and observing the TOE reject the password) and of 8 characters (and observing that the TOE accepted the password change). The evaluator also confirmed that a password one larger than the maximum claimed by [ST] was rejected.

2.5.3 Pre-Shared Key Composition - per TD0838 (VPNGW13:FIA_PSK_EXT.1)

2.5.3.1 VPNGW13:FIA_PSK_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.3.2 VPNGW13:FIA_PSK_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure that it identifies all protocols that allow pre-shared keys. For each protocol identified by the requirement, the evaluator shall confirm that the TSS states which pre-shared key selections are supported.

Section 6.1, Table 19 of [ST] provides the following:

FIA_PSK_EXT.1 [VPN] states that the TOE supports use of IKEv2 pre-shared keys for authentication of IPsec tunnels. Pre-shared keys can be entered as ASCII character strings, or HEX values.

FCS_IPSEC_EXT.1 states the text-based pre-shared keys can be composed of any combination of upper and lower case letters, numbers, and special characters. The TOE supports keys that are from 1 character in length up to 128 characters in length. The text-based pre-shared key is conditioned by one of the prf functions (HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384 or HMAC-SHA-512) configured by the administrator.

FCS_IPSEC_EXT.1 also states that pre-shared keys can be configured in the TOE for IPsec connection authentication when using IKEv2 for peer-to-peer VPNs. The bit-based pre-shared keys can be entered as HEX value as well. When using pre-shared keys for authentication, the IPsec endpoints must both be configured to use the same key.

Component Guidance Assurance Activities: The evaluator shall examine the operational guidance to determine that it provides guidance to administrators on how to configure all selected pre-shared key options if any configuration is required.

The evaluator shall examine the operational guidance to determine that it provides guidance to administrators on how to configure the mandatory_or_not flag per RFC 8784. (TD0838 applied)

The “Post Quantum Preshared Key (PPK) Configuration” section of the [AdminGuide] indicates that Post Quantum Preshared Keys (PPKs) are included in the IKEv2 key material to make the exchange secure from quantum attacks. This section provides the commands that are used to configure Post Quantum Preshared Key (PPK) compliance with RFC 8784.

This section also explains that the *Mandatory Flag*, configures the PPK as mandatory for the connection. This can be added after the *identifier*. Adding this flag marks adding the PPK as mandatory.

Component Testing Assurance Activities: The evaluator shall also perform the following tests for each protocol (or instantiation of a protocol, if performed by a different implementation on the TOE).

Test 1: For each mechanism selected in FIA_PSK_EXT.1.2 the evaluator shall attempt to establish a connection and confirm that the connection requires the selected factors in the PSK to establish the connection in alignment with table 1 from RFC 8784. (TD0838 applied)

The evaluator configured the TOE and a test server to use generated bit-based pre-shared keys for authentication during IPsec IKEv2 negotiations. Using this configuration, the evaluator was able to establish an IPsec connection between the TOE and the IPsec test peer. Generated bit-based pre-shared keys is the only mechanism selected in

FIA_PSK_EXT.1.2. During this test the evaluator additionally configured a Postquantum Pre-shared Key (PPK) on the TOE and observed that it was able to be used in a connection to confirm the TOE's ability to comply with RFC 8784.

2.5.4 Generated Pre-Shared Keys (VPNGW13:FIA_PSK_EXT.2)

2.5.4.1 VPNGW13:FIA_PSK_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: If 'generate' is selected, the evaluator shall confirm that this process uses the RBG specified in FCS_RBG_EXT.1 and the output matches the size selected in FIA_PSK_EXT.2.1.

Not applicable, 'generate' is not selected in the [ST].

Component Guidance Assurance Activities: The evaluator shall confirm the operational guidance contains instructions for entering generated pre-shared keys for each protocol identified in the FIA_PSK_EXT.1.1.

The "Post Quantum Preshared Key (PPK) Configuration" section of the [AdminGuide] indicates that Post Quantum Preshared Keys (PPKs) are included in the IKEv2 key material to make the exchange secure from quantum attacks. This section provides the commands that are used to configure Post Quantum Preshared Key (PPK) compliance with RFC 8784.

This section also explains that the *Mandatory Flag*, configures the PPK as mandatory for the connection. This can be added after the *identifier*. Adding this flag marks adding the PPK as mandatory.

Component Testing Assurance Activities: Test 1: [conditional] If generate was selected the evaluator shall generate a pre-shared key and confirm the output matches the size selected in FIA_PSK_EXT.2.1.

Not applicable. The TOE does not generate pre-shared keys.

2.5.5 Protected Authentication Feedback (NDcPP22e:FIA_UAU.7)

2.5.5.1 NDcPP22e:FIA_UAU.7.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: The evaluator shall examine the guidance documentation to determine that any necessary preparatory steps to ensure authentication data is not revealed while entering for each local login allowed.

The TOE always protects authentication data as it is entered, no administrative actions are needed for this feature.

Component Testing Assurance Activities: The evaluator shall perform the following test for each method of local login allowed:

a) Test 1: The evaluator shall locally authenticate to the TOE. While making this attempt, the evaluator shall verify that at most obscured feedback is provided while entering the authentication information.

The evaluator observed that passwords are obscured while logging in at the console and via SSH.

2.5.6 Password-based Authentication Mechanism (NDcPP22e:FIA_UAU_EXT.2)

2.5.6.1 NDcPP22e:FIA_UAU_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: Evaluation Activities for this requirement are covered under those for FIA_UIA_EXT.1. If other authentication mechanisms are specified, the evaluator shall include those methods in the activities for FIA_UIA_EXT.1.

See FIA_UIA_EXT.1.

Component Guidance Assurance Activities: Evaluation Activities for this requirement are covered under those for FIA_UIA_EXT.1. If other authentication mechanisms are specified, the evaluator shall include those methods in the activities for FIA_UIA_EXT.1.

See FIA_UIA_EXT.1.

Component Testing Assurance Activities: Evaluation Activities for this requirement are covered under those for FIA_UIA_EXT.1. If other authentication mechanisms are specified, the evaluator shall include those methods in the activities for FIA_UIA_EXT.1.

See NDcPP22e:FIA_UIA_EXT.1.

2.5.7 User Identification and Authentication (NDcPP22e:FIA_UIA_EXT.1)

2.5.7.1 NDcPP22e:FIA_UIA_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.7.2 NDcPP22e:FIA_UIA_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it describes the logon process for each logon method (local, remote (HTTPS, SSH, etc.)) supported for the product. This description shall contain information pertaining to the credentials allowed/used, any protocol transactions that take place, and what constitutes a 'successful logon'.

The evaluator shall examine the TSS to determine that it describes which actions are allowed before user identification and authentication. The description shall cover authentication and identification for local and remote TOE administration.

For distributed TOEs the evaluator shall examine that the TSS details how Security Administrators are authenticated and identified by all TOE components. If not all TOE components support authentication of Security Administrators according to FIA_UIA_EXT.1 and FIA_UAU_EXT.2, the TSS shall describe how the overall TOE functionality is split between TOE components including how it is ensured that no unauthorized access to any TOE component can occur.

For distributed TOEs, the evaluator shall examine the TSS to determine that it describes for each TOE component which actions are allowed before user identification and authentication. The description shall cover authentication and identification for local and remote TOE administration. For each TOE component that does not support authentication of Security Administrators according to FIA_UIA_EXT.1 and FIA_UAU_EXT.2 the TSS shall describe any unauthenticated services/services that are supported by the component.

Section 6.1, Table 19 of [ST] provides the following:

FIA_UIA_EXT.1 states that the TOE requires all users to be successfully identified and authenticated before allowing any TSF mediated actions to be performed. Administrative access to the TOE is facilitated through the TOE's CLI (SSH or console), and through the GUI (ASDM). The TOE mediates all administrative actions through the CLI and GUI. Once a potential administrative user attempts to access an administrative interface either locally or remotely, the TOE prompts the user for a user name and password. Only after the administrative user presents the correct authentication credentials will access to the TOE administrative functionality be granted. No access is allowed to the administrative functionality of the TOE until an administrator is successfully identified and

authenticated. The TOE also supports authentication via SSH public key. Administrators can login to the TOE using SSH keys which are provided during the SSH connection request.

FIA_UAU_EXT.2 states that the TOE provides a local password-based authentication mechanism as well as RADIUS and TACACS+ authentication. The administrator authentication policies include authentication to the local user database or redirection to a remote authentication server. Interfaces can be configured to try one or more remote authentication servers, and then fall back to the local user database if the remote authentication servers are inaccessible.

The TOE can invoke an external authentication server to provide a single-use authentication mechanism by forwarding the authentication requests to the external authentication server (when configured by the TOE to provide single-use authentication).

The process for authentication is the same for administrative access whether administration is occurring via a directly connected console cable or remotely via SSHv2 or TLS.

FTA_TAB.1 states that the TOE provides administrators with the capability to configure an advisory banner or warning message(s) that is displayed prior to completion of the logon process at the local console or via any remote connection (e.g., SSH or HTTPS).

The TOE is not distributed.

Component Guidance Assurance Activities: The evaluator shall examine the guidance documentation to determine that any necessary preparatory steps (e.g., establishing credential material such as pre-shared keys, tunnels, certificates, etc.) to logging in are described. For each supported the login method, the evaluator shall ensure the guidance documentation provides clear instructions for successfully logging on. If configuration is necessary to ensure the services provided before login are limited, the evaluator shall determine that the guidance documentation provides sufficient instruction on limiting the allowed services.

The “Administrative Access” section of the [AdminGuide] describes the three methods by which the administrator can manage the security appliance: Console (serial port), SSH and TLS. It notes that Telnet is not permitted for management on the TOE and must remain in its default state which is disabled. The “Administration” section of the [AdminGuide] describes the security functionality that an administrator can perform via the local console, remotely via SSH or remotely via TLS (on the ASDM). This section also explains that optionally, the TOE supports tunneling the ASDM and/or SSH connections in IPsec VPN tunnels (peer-to-peer, or remote VPN client).

The “Enabling HTTPS Access” section of the [AdminGuide] provides instructions for configuring the HTTPS server to allow HTTPS connections to the TOE in order to access the ASDM. The “Accessing ASDM From your Workstation” section of the [AdminGuide] describes how to access the TOE via a web browser.

The “Configuring SSH” section of the [AdminGuide] provides instructions for configuring inbound SSH sessions for remote administration of the ASA.

The “Configuring TLS” and “Configuring IPsec” sections in the [AdminGuide] provide detailed instructions for configuring TLS and IPsec options.

The “Passwords” section of the [AdminGuide] provides guidance on configuring password complexity and minimum length. The “Using Pre-Shared Keys” section of the [AdminGuide] describes how to use the preshared key authentication method and it provides guidance to the user for creating complex strong keys.

The “Local and Remote Access to ASA” section of the [AdminGuide] describes how administrative access to any administrative interface (console and/or HTTPS/TLS) can be configured to use remote AAA (RADIUS) authentication, and/or local authentication.

The “Login Banners” section of the [AdminGuide] provides instructions for configuring login banners for the CLI (console and SSH access) as well as for the GUI (ASDM).

Component Testing Assurance Activities: The evaluator shall perform the following tests for each method by which administrators access the TOE (local and remote), as well as for each type of credential supported by the login method:

a) Test 1: The evaluator shall use the guidance documentation to configure the appropriate credential supported for the login method. For that credential/login method, the evaluator shall show that providing correct I&A information results in the ability to access the system, while providing incorrect information results in denial of access.

b) Test 2: The evaluator shall configure the services allowed (if any) according to the guidance documentation, and then determine the services available to an external remote entity. The evaluator shall determine that the list of services available is limited to those specified in the requirement.

c) Test 3: For local access, the evaluator shall determine what services are available to a local administrator prior to logging in, and make sure this list is consistent with the requirement.

d) Test 4: For distributed TOEs where not all TOE components support the authentication of Security Administrators according to FIA_UIA_EXT.1 and FIA_UAU_EXT.2, the evaluator shall test that the components authenticate Security Administrators as described in the TSS.

The TOE offers the following user interfaces where authentication is provided.

- Local ASA Console
- SSH to ASA using passwords
- SSH to ASA using public/private key pairs
- SSH Connection using Radius Account
- SSH Connection using TACACS+ Account
- HTTPS (ASDM) using passwords

Test 1 - Using each interface the evaluator performed an unsuccessful and successful logon of each type using bad and good credentials respectively.

Test 2 – Using an Nmap scan, the evaluator determined which services were available to an external remote entity. The list of available services was confirmed by the evaluator to be limited to those specified in the requirement.

Test 3 - Using each interface the evaluator found that, prior to login, no functions were available to the administrator with the exception of acknowledging the banner.

Test 4 - The TOE is not distributed, thus tests 1 through 3 above test the only TOE component.

2.5.8 X.509 Certificate Validation (NDcPP22e:FIA_X509_EXT.1/Rev)

2.5.8.1 NDcPP22e:FIA_X509_EXT.1.1/Rev

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall demonstrate that checking the validity of a certificate is performed when a certificate is used in an authentication step or when performing trusted updates (if FPT_TUD_EXT.2 is selected). It is not sufficient to verify the status of a X.509 certificate only when it is loaded onto the TOE. It is not necessary to verify the revocation status of X.509 certificates during power-up self-tests (if the option for using X.509 certificates for self-testing is selected). The evaluator shall perform the following tests for FIA_X509_EXT.1.1/Rev. These tests must be repeated for each distinct security function that utilizes X.509v3 certificates. For example, if the TOE implements certificate-based authentication with IPSEC and TLS, then it shall be tested with each of these protocols:

a) Test 1a: The evaluator shall present the TOE with a valid chain of certificates (terminating in a trusted CA certificate) as needed to validate the leaf certificate to be used in the function, and shall use this chain to demonstrate that the function succeeds. Test 1a shall be designed in a way that the chain can be 'broken' in Test 1b by either being able to remove the trust anchor from the TOEs trust store, or by setting up the trust store in a way that at least one intermediate CA certificate needs to be provided, together with the leaf certificate from outside the TOE, to complete the chain (e.g. by storing only the root CA certificate in the trust store).

Test 1b: The evaluator shall then 'break' the chain used in Test 1a by either removing the trust anchor in the TOE's trust store used to terminate the chain, or by removing one of the intermediate CA certificates (provided together with the leaf certificate in Test 1a) to complete the chain. The evaluator shall show that an attempt to validate this broken chain fails.

b) Test 2: The evaluator shall demonstrate that validating an expired certificate results in the function failing.

c) Test 3: The evaluator shall test that the TOE can properly handle revoked certificates - conditional on whether CRL or OCSP is selected; if both are selected, then a test shall be performed for each method. The evaluator shall test revocation of the peer certificate and revocation of the peer intermediate CA certificate i.e. the intermediate CA certificate should be revoked by the root CA. The evaluator shall ensure that a valid certificate is used, and that

the validation function succeeds. The evaluator then attempts the test with a certificate that has been revoked (for each method chosen in the selection) to ensure when the certificate is no longer valid that the validation function fails. Revocation checking is only applied to certificates that are not designated as trust anchors. Therefore the revoked certificate(s) used for testing shall not be a trust anchor.

d) Test 4: If OCSP is selected, the evaluator shall configure the OCSP server or use a man-in-the-middle tool to present a certificate that does not have the OCSP signing purpose and verify that validation of the OCSP response fails. If CRL is selected, the evaluator shall configure the CA to sign a CRL with a certificate that does not have the cRLsign key usage bit set, and verify that validation of the CRL fails.

e) Test 5: The evaluator shall modify any byte in the first eight bytes of the certificate and demonstrate that the certificate fails to validate. (The certificate will fail to parse correctly.)

f) Test 6: The evaluator shall modify any byte in the last byte of the certificate and demonstrate that the certificate fails to validate. (The signature on the certificate will not validate.)

g) Test 7: The evaluator shall modify any byte in the public key of the certificate and demonstrate that the certificate fails to validate. (The hash of the certificate will not validate.)

h) The following tests are run when a minimum certificate path length of three certificates is implemented.

Test 8: (Conditional on support for EC certificates as indicated in FCS_COP.1/SigGen). The evaluator shall conduct the following tests:

Test 8a: (Conditional on TOE ability to process CA certificates presented in certificate message) The test shall be designed in a way such that only the EC root certificate is designated as a trust anchor, and by setting up the trust store in a way that the EC Intermediate CA certificate needs to be provided, together with the leaf certificate, from outside the TOE to complete the chain (e.g. by storing only the EC root CA certificate in the trust store). The evaluator shall present the TOE with a valid chain of EC certificates (terminating in a trusted CA certificate), where the elliptic curve parameters are specified as a named curve. The evaluator shall confirm that the TOE validates the certificate chain.

Test 8b: (Conditional on TOE ability to process CA certificates presented in certificate message) The test shall be designed in a way such that only the EC root certificate is designated as a trust anchor, and by setting up the trust store in a way that the EC Intermediate CA certificate needs to be provided, together with the leaf certificate, from outside the TOE to complete the chain (e.g. by storing only the EC root CA certificate in the trust store). The evaluator shall present the TOE with a chain of EC certificates (terminating in a trusted CA certificate), where the intermediate certificate in the certificate chain uses an explicit format version of the Elliptic Curve parameters in the public key information field, and is signed by the trusted EC root CA, but having no other changes. The evaluator shall confirm the TOE treats the certificate as invalid.

Test 8c: The evaluator shall establish a subordinate CA certificate, where the elliptic curve parameters are specified as a named curve, that is signed by a trusted EC root CA. The evaluator shall attempt to load the certificate into the trust store and observe that it is accepted into the TOE's trust store. The evaluator shall then establish a

subordinate CA certificate that uses an explicit format version of the elliptic curve parameters, and that is signed by a trusted EC root CA. The evaluator shall attempt to load the certificate into the trust store and observe that it is rejected, and not added to the TOE's trust store.

(TD0527 12/2020 update applied)

These tests were performed for Syslog Cert Validation and IPsec Cert Validation.

Test 1a & 1b -- The evaluator configured the TOE and a peer with valid certificates. The evaluator then attempted to make a connection between the peer devices. A successful connection was made. The evaluator then configured a server certificate with an invalid certification path by deleting the root CA so that the certificate chain was invalid because of a missing (or deleted) certificate. The connection between the peers was refused.

Test 2 -- The evaluator attempted to make a connection between the TOE and a test server. The test server then presented an expired certificate during the negotiation resulting in a failed connection.

Test 3 -- The evaluator attempted to make a connection between the TOE and a test server using TLS and repeated using IPsec. The test server then presented a certificate during the protocol negotiation where the certificate was valid. A packet capture was obtained of these protocol negotiations which shows that the connection was successful. The evaluator revoked certificates in the chain (individually) and attempted the same connection. The attempt after revoking the certificate was not successful.

Test 4 -- The evaluator configured an OCSP responder to present a certificate that does not have the OCSP signing purpose. The evaluator established a connection between the TOE and a test server such that the TOE receives OCSP response signed by the invalid certificate and ensured that the connection was not negotiated successfully. The evaluator also configured the TOE to use only CRL and established a connection between the TOE and the test server such that the TOE receives CRL signed by the invalid certificate (that does not have a cRLsign key usage bit) and ensured that the connection was not negotiated successfully.

Test 5 -- The evaluator configured a test server to present a certificate that had a byte in the first eight bytes modified to the TOE. The evaluator then attempted to make a connection between the peer devices. When the TOE attempted to connect, the connection failed.

Test 6 -- The evaluator configured a test server to present a certificate that had a byte in the last eight bytes modified to the TOE. The evaluator then attempted to make a connection between the peer devices. When the TOE attempted to connect, the connection failed.

Test 7 -- The evaluator configured a test server to present a certificate that had a byte in the public key of the certificate modified to the TOE. The evaluator then attempted to make a connection between the peer devices. When the TOE attempted to connect, the connection was refused.

Test 8a - The evaluator configured the test peer to send a valid chain of ECDSA certificates where only the root CA is located in the trust store, and where the ECDSA certificates have a supported named curve. The TOE accepted the certificate as valid.

Test 8b - The evaluator then configured the test peer to send a valid chain of ECDSA certificates where only the root CA is located in the trust store, and where an intermediate CA's ECDSA certificate has an explicit curve. The TOE rejected the certificate as invalid.

Test 8c - The evaluator attempted to load an intermediate CA ECDSA certificate with a supported named curve into the TOE trust store and observed that the certificate can be loaded. The evaluator attempted to load an intermediate CA ECDSA certificate with an explicit curve into the TOE trust store and observed that the certificate could not be loaded.

2.5.8.2 NDcPP22e:FIA_X509_EXT.1.2/Rev

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall perform the following tests for FIA_X509_EXT.1.2/Rev. The tests described must be performed in conjunction with the other certificate services assurance activities, including the functions in FIA_X509_EXT.2.1/Rev. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules. Where the TSS identifies any of the rules for extendedKeyUsage fields (in FIA_X509_EXT.1.1) that are not supported by the TOE (i.e. where the ST is therefore claiming that they are trivially satisfied) then the associated extendedKeyUsage rule testing may be omitted.

The goal of the following tests is to verify that the TOE accepts a certificate as a CA certificate only if it has been marked as a CA certificate by using basicConstraints with the CA flag set to True (and implicitly tests that the TOE correctly parses the basicConstraints extension as part of X509v3 certificate chain validation). For each of the following tests the evaluator shall create a chain of at least three certificates: a self-signed root CA certificate, an intermediate CA certificate and a leaf (node) certificate. The properties of the certificates in the chain are adjusted as described in each individual test below (and this modification shall be the only invalid aspect of the relevant certificate chain).

a) Test 1: The evaluator shall ensure that at least one of the CAs in the chain does not contain the basicConstraints extension. The evaluator confirms that the TOE rejects such a certificate at one (or both) of the following points: (i) as part of the validation of the leaf certificate belonging to this chain; (ii) when attempting to add a CA certificate without the basicConstraints extension to the TOE's trust store (i.e. when attempting to install the CA certificate as one which will be retrieved from the TOE itself when validating future certificate chains).

b) Test 2: The evaluator shall ensure that at least one of the CA certificates in the chain has a basicConstraints extension in which the CA flag is set to FALSE. The evaluator confirms that the TOE rejects such a certificate at one (or both) of the following points: (i) as part of the validation of the leaf certificate belonging to this chain; (ii) when attempting to add a CA certificate with the CA flag set to FALSE to the TOE's trust store (i.e. when attempting to install the CA certificate as one which will be retrieved from the TOE itself when validating future certificate chains).

The evaluator shall repeat these tests for each distinct use of certificates. Thus, for example, use of certificates for TLS connection is distinct from use of certificates for trusted updates so both of these uses would be tested. But there is no need to repeat the tests for each separate TLS channel in FTP_ITC.1 and FTP_TRP.1/Admin (unless the channels use separate implementations of TLS).

Test 1: The evaluator configured a test syslog server to present a certificate chain containing a CA certificate lacking the basicConstraints extension. The evaluator then used the TOE TLS client (syslog) to attempt to connect to the test server. The evaluator observed that the TOE rejected the connections.

Test 2: The evaluator configured a test server to present a certificate chain containing a CA certificate having the basicConstraints section but with the cA flag not set (i.e., FALSE). The evaluator then used the TOE TLS client (syslog) to attempt to connect to the test server. The evaluator observed that the TOE rejected the connection.

Component TSS Assurance Activities: The evaluator shall ensure the TSS describes where the check of validity of the certificates takes place, and that the TSS identifies any of the rules for extendedKeyUsage fields (in FIA_X509_EXT.1.1) that are not supported by the TOE (i.e. where the ST is therefore claiming that they are trivially satisfied). It is expected that revocation checking is performed when a certificate is used in an authentication step and when performing trusted updates (if selected). It is not necessary to verify the revocation status of X.509 certificates during power-up self-tests (if the option for using X.509 certificates for self-testing is selected).

The TSS shall describe when revocation checking is performed and on what certificates. If the revocation checking during authentication is handled differently depending on whether a full certificate chain or only a leaf certificate is being presented, any differences must be summarized in the TSS section and explained in the Guidance.

Section 6.1, Table 19 (FIA_X509_EXT.1/Rev) of [ST] states that the TOE uses X.509v3 certificates as defined by RFC 5280 to support authentication for TLS and IPsec connections.

The validity check for the certificates takes place at session establishment and/or at time of import depending on the certificate type. For example, server certificate is checked at session establishment while CA certificate is checked at both. The TOE conforms to standard RFC 5280 for certificate and path validation.

The TOE generates an RSA or ECDSA key pair that can be embedded in a Certificate Signing Request (CSR) created by the TOE. The TOE then sends the CSR manually to a Certificate Authority (CA) for the CA to sign and issue a certificate. Once the certificate has been issued, the administrator can import the X.509v3 certificate into the TOE. Integrity of the CSR and certificate during transit are assured through the use of digital signature (signing the hash of the TOE's public key contained in the CSR and certificate). Both OCSP and CRL are configurable and may be used for certificate revocation checking when the TOE is validating server certificates when initiating outbound TLS connections to syslog and AAA servers and for IPsec connections. Checking is also done for the basicConstraints extension and the cA flag to determine whether they are present and set to TRUE. If they are not, the CA certificate is not accepted as a trustpoint. In all use cases (whether using CRL or OCSP) if the connection to determine the certificate validity cannot be established, the TOE will not accept the certificate.

Component Guidance Assurance Activities: The evaluator shall also ensure that the guidance documentation describes where the check of validity of the certificates takes place, describes any of the rules for extendedKeyUsage fields (in FIA_X509_EXT.1.1) that are not supported by the TOE (i.e. where the ST is therefore claiming that they are trivially satisfied) and describes how certificate revocation checking is performed and on which certificate.

The section entitled "Using X.509v3 Digital Certificates" in [AdminGuide] provides a detailed description of X.509v3 Digital Certificates and how they are used by ASA. This section explains that X.509v3 certificates as defined by RFC 5280 are used for authentication of a network peer using IPsec and TLS.

This section states that digital certificates provide digital identification for authentication. A digital certificate includes information that identifies a device or user, such as the common name, serial number, company, department, state, country, or IP address. CAs are trusted authorities that "sign" certificates to verify their authenticity, thereby guaranteeing the identity of the device or user. CAs issue digital certificates in the context of a PKI, which uses public-key or private-key encryption to ensure security.

This section also explains that an identity certificate intended to authenticate a TLS server or TLS client will contain a "*ServerAuth*" or "*ClientAuth*" Extended key Usage (EKU), while one intended for use by peers in an IPsec VPN would contain an "*IPsec Tunnel*" EKU. ASA does not enforce any other EKU.

Finally, this section explains that certificate revocation checking occurs on all certificates except self-signed Root Certificate Authorities when either CRL or OCSP `revocation-check` has been defined for a trustpoint.

Component Testing Assurance Activities: None Defined

2.5.9 X.509 Certificate Validation (VPNGW13:FIA_X509_EXT.1/Rev)

2.5.9.1 VPNGW13:FIA_X509_EXT.1.1/Rev

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: There is no change to the Evaluation Activities specified for this SFR in the NDcPP Supporting Document. The PP-Module modifies this SFR to make it mandatory because of the TOE's required support for IPsec.

There is no change to the Evaluation activities specified for this SFR from those in the NDcPP Supporting document. See NDcPP22e:FIA_X509_EXT.1/Rev.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.5.10 X.509 Certificate Authentication (NDcPP22e:FIA_X509_EXT.2)

2.5.10.1 NDcPP22e:FIA_X509_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.10.2 NDcPP22e:FIA_X509_EXT.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check the TSS to ensure that it describes how the TOE chooses which certificates to use, and any necessary instructions in the administrative guidance for configuring the operating environment so that the TOE can use the certificates.

The evaluator shall examine the TSS to confirm that it describes the behaviour of the TOE when a connection cannot be established during the validity check of a certificate used in establishing a trusted channel. The evaluator shall verify that any distinctions between trusted channels are described. If the requirement that the administrator is able to specify the default action, then the evaluator shall ensure that the guidance documentation contains instructions on how this configuration action is performed.

Section 6.1, Table 19 (FIA_X509_EXT.2) in [ST] explains that administrators can configure a trustpoint and associate it with a crypto map. This will tell the TOE which certificate(s) to use during the validation process. When the TOE cannot establish a connection for the validity check (whether using CRL or OCSP), the trusted channel is not established.

Component Guidance Assurance Activities: The evaluator shall also ensure that the guidance documentation describes the configuration required in the operating environment so the TOE can use the certificates. The guidance documentation shall also include any required configuration on the TOE to use the certificates. The guidance document shall also describe the steps for the Security Administrator to follow if the connection cannot be established during the validity check of a certificate used in establishing a trusted channel.

The section entitled "Using X.509v3 Digital Certificates" and “Specify the TLS Server and Client Certificates” in [AdminGuide] explain that network peers in the operational environment to which the ASA will connect using TLS or IPsec, must be configured to present a valid x509v3 identity certificate issued by a PKI trusted by the ASA. For example, if audit transfer is protected by TLS, then the TLS connection offered by the audit server must provide a valid x509v3 identity certificate.

The section entitled "Create Trustpoint and Generate Certificate Signing Request (CSR)" in [AdminGuide] provides instructions on how to: create and configure a CA trustpoint, import the CA certificate for the configured trustpoint, create certificate signing requests on the TOE which can be used to obtain an X509v3 identity certificate and import the subsequent certificates generated from certificate requests.

The section entitled "Using X.509v3 Digital Certificates" and “Specify the TLS Server and Client Certificates” state that when the TOE cannot establish a connection for the validity check using CRL or the OCSP responder for verification, the TOE will not accept the certificate and the trusted channel will not be established. If a TLS or IPsec session fails due to inability to contact the CRL server or OCSP server, the connectivity to the CRL or OCSP server should be restored before reattempting to establish the session.

The sections “CRLs” and “OCSP” sections describe how administrator can configure the ASA to make CRL or OCSP checks mandatory when authenticating a certificate by using the ***revocation-check crl*** or ***revocation-check ocsp*** commands.

Component Testing Assurance Activities: The evaluator shall perform the following test for each trusted channel:

The evaluator shall demonstrate that using a valid certificate that requires certificate validation checking to be performed in at least some part by communicating with a non-TOE IT entity. The evaluator shall then manipulate the environment so that the TOE is unable to verify the validity of the certificate, and observe that the action selected in FIA_X509_EXT.2.2 is performed. If the selected action is administrator-configurable, then the evaluator shall follow the guidance documentation to determine that all supported administrator-configurable options behave in their documented manner.

The evaluator established a syslog connection from the TOE to a remote test server protected by TLS and obtained a packet capture of the activity. This was repeated when the OCSP responder was available and unavailable. When available the connection succeeded. When the OCSP responder was unavailable, the connection failed.

2.5.11 X.509 Certificate Authentication (VPNGW13:FIA_X509_EXT.2)

2.5.11.1 VPNGW13:FIA_X509_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.11.2 VPNGW13:FIA_X509_EXT.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: There is no change to the Evaluation Activities specified for this SFR in the NDcPP Supporting Document. The PP-Module modifies this SFR to support its use for IPsec at a minimum. The evaluator shall ensure that all evaluation of this SFR is performed against its use in IPsec communications as well as any other supported usage.

There is no change to the Evaluation activities specified for this SFR in the NDcPP Supporting document. See NDcPP22e:FIA_X509_EXT.2 which demonstrates that IPsec communications were included in the evaluation of this SFR.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.5.12 X.509 Certificate Requests (NDcPP22e:FIA_X509_EXT.3)

2.5.12.1 NDcPP22e:FIA_X509_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.12.2 NDcPP22e:FIA_X509_EXT.3.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: If the ST author selects 'device-specific information', the evaluator shall verify that the TSS contains a description of the device-specific fields used in certificate requests.

[ST] does not include the selection 'device-specific information'.

Component Guidance Assurance Activities: The evaluator shall check to ensure that the guidance documentation contains instructions on requesting certificates from a CA, including generation of a Certification Request. If the ST author selects 'Common Name', 'Organization', 'Organizational Unit', or 'Country', the evaluator shall ensure that this guidance includes instructions for establishing these fields before creating the Certification Request.

The “Using X.509v3 Digital Certificates” section of the [AdminGuide] describes how the TOE supports X.509v3 certificates as defined by RFC 5280.

The “Create Trustpoint and Generate Certificate Signing Request (CSR)” section of the [AdminGuide] describes how to generate an RSA or ECDSA key pair, create and configure a CA trustpoint, import the CA certificate for the configured trustpoint, generate the CSR and import the certificate to the TOE. Prior to creating the certificate

request message, the administrator will create and configure a CA trustpoint using the 'subject-name' command which specifies the Common Name (CN), Organizational Unit (OU), Organization (O) and Country (C).

Component Testing Assurance Activities: The evaluator shall perform the following tests:

a) Test 1: The evaluator shall use the guidance documentation to cause the TOE to generate a Certification Request. The evaluator shall capture the generated request and ensure that it conforms to the format specified. The evaluator shall confirm that the Certification Request provides the public key and other required information, including any necessary user-input information.

b) Test 2: The evaluator shall demonstrate that validating a response message to a Certification Request without a valid certification path results in the function failing. The evaluator shall then load a certificate or certificates as trusted CAs needed to validate the response message, and demonstrate that the function succeeds.

Test 1- The evaluator generated a certificate signing request by following the instructions in the administration guidance for generating the request. The evaluator viewed the contents of the CSR and confirmed that it provided public key and other required information including the information that the evaluator supplied during the generation process.

Test 2 - The evaluator attempted to import a certificate that was signed by a root CA that was not installed on the TOE. The attempt failed. The evaluator then successfully installed the certificate resulting from the CSR request in test 1.

2.5.13 X.509 Certificate Requests (VPNGW13:FIA_X509_EXT.3)

2.5.13.1 VPNGW13:FIA_X509_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.13.2 VPNGW13:FIA_X509_EXT.3.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: There is no change to the Evaluation Activities specified for this SFR in the NDcPP Supporting Document. The PP-Module modifies this SFR to make it mandatory because of the TOE's required support for IPsec.

There is no change to the Evaluation Activities specified for this SFR in the NDcPP Supporting document. See NDCPP22e:FIA_X509_EXT.3.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.6 Security management (FMT)

2.6.1 Management of security functions behaviour (NDcPP22e:FMT_MOF.1/ManualUpdate)

2.6.1.1 NDcPP22e:FMT_MOF.1.1/ManualUpdate

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: For distributed TOEs it is required to verify the TSS to ensure that it describes how every function related to security management is realized for every TOE component and shared between different TOE components. The evaluator shall confirm that all relevant aspects of each TOE component are covered by the FMT SFRs.

There are no specific requirements for non-distributed TOEs.

Not applicable as the TOE is not a distributed TOE.

Component Guidance Assurance Activities: The evaluator shall examine the guidance documentation to determine that any necessary steps to perform manual update are described. The guidance documentation shall also provide warnings regarding functions that may cease to operate during the update (if applicable).

For distributed TOEs the guidance documentation shall describe all steps how to update all TOE components. This shall contain description of the order in which components need to be updated if the order is relevant to the update process. The guidance documentation shall also provide warnings regarding functions of TOE components and the overall TOE that may cease to operate during the update (if applicable).

The “Software (ASA) Installation” section of the [AdminGuide] describes the verification and update process. This includes downloading the evaluated software image file from Cisco.com onto a trusted computer system and verifying the downloaded image using the “verify” command. The digital signature uses 2048 bit RSA with SHA 512.

Component Testing Assurance Activities: The evaluator shall try to perform the update using a legitimate update image without prior authentication as Security Administrator (either by authentication as a user with no administrator privileges or without user authentication at all - depending on the configuration of the TOE). The attempt to update the TOE should fail.

The evaluator shall try to perform the update with prior authentication as Security Administrator using a legitimate update image. This attempt should be successful. This test case should be covered by the tests for FPT_TUD_EXT.1 already.

The operations to perform an update of the TOE are available to all users that can login to the TOE, because only administrators can login. The set of functions available to a user prior to login do not include TOE update as specified by NDcPP22e:FIA_UIA_EXT.1.

As can be seen in the NDCPP22e: FIA_UIA_EXT.1-t3 test evidence, no functions are offered to users prior to a successful login. Any user that can login, is considered an administrator and can perform TOE updates. Testing activities for FPT_TUD_EXT.1 – Test 1 demonstrate a successful update attempt performed by a Security Administrator with prior authentication.

2.6.2 Management of TSF Data (NDcPP22e:FMT_MTD.1/CoreData)

2.6.2.1 NDcPP22e:FMT_MTD.1.1/CoreData

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that, for each administrative function identified in the guidance documentation; those that are accessible through an interface prior to administrator log-in are identified. For each of these functions, the evaluator shall also confirm that the TSS details how the ability to manipulate the TSF data through these interfaces is disallowed for non-administrative users.

If the TOE supports handling of X.509v3 certificates and implements a trust store, the evaluator shall examine the TSS to determine that it contains sufficient information to describe how the ability to manage the TOE's trust store is restricted.

Section 6.1, Table 19 of [ST] provides the following:

FIA_UIA_EXT.1 states that the TOE requires all users to be successfully identified and authenticated before allowing any TSF mediated actions to be performed. No access is allowed to the administrative functionality of the TOE until an administrator is successfully identified and authenticated.

FMT_SMF.1 states that the TOE does not provide services (other than connecting using HTTPS, SSH and establishment of VPNs) prior to authentication.

FMT_MTD.1/CoreData explains that the TOE provides the ability for authorized administrators to access TOE data, such as audit data, configuration data, security attributes, routing tables, and session thresholds. The TOE also restricts access to TSF data so that no manipulation can be performed by non-administrators. Each of the

predefined and administratively configured privilege level has default set of permissions that will grant them access to the TOE data, though with some privilege levels, the access is limited.

FIA_X509_EXT.1/Rev states that administrators can configure a trustpoint. The evaluator noted that since the set of configured trustpoints represent a trust store, the limitation of trustpoint configuration to administrators defines the TOE's ability to manage its trust store.

Component Guidance Assurance Activities: The evaluator shall review the guidance documentation to determine that each of the TSF-data-manipulating functions implemented in response to the requirements of the cPP is identified, and that configuration information is provided to ensure that only administrators have access to the functions.

If the TOE supports handling of X.509v3 certificates and provides a trust store, the evaluator shall review the guidance documentation to determine that it provides sufficient information for the administrator to configure and maintain the trust store in a secure way. If the TOE supports loading of CA certificates, the evaluator shall review the guidance documentation to determine that it provides sufficient information for the administrator to securely load CA certificates into the trust store. The evaluator shall also review the guidance documentation to determine that it explains how to designate a CA certificate a trust anchor.

The TSF data manipulating functions and the corresponding configuration information from specific sections of the [AdminGuide] are identified or referenced throughout this AAR with the requirement to which they apply.

The “Usernames, Privileges, and Administrative Roles” section of the [AdminGuide] describes how usernames and privilege levels are configured. The privilege level determines the functions the user can perform; hence the authorized administrator with the appropriate privileges. The authorized administrator in the TOE is a privilege level 15 user. Users with the authorized administrator role have access to all TOE security functions including importing X.509v3 certificates to the TOE’s trust store.

See NDcPP22e:FIA_X509_EXT.2.2 which identifies the sections in the Guide which describe how the administrator role can configure and maintain the trust store including loading of CA certificates.

Component Testing Assurance Activities: No separate testing for FMT_MTD.1/CoreData is required unless one of the management functions has not already been exercised under any other SFR.

All of the management functions have been exercised under the other SFRs as demonstrated throughout the AAR.

2.6.3 Management of TSF Data (NDcPP22e:FMT_MTD.1/CryptoKeys)

2.6.3.1 NDcPP22e:FMT_MTD.1.1/CryptoKeys

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: For distributed TOEs see chapter 2.4.1.1.

For non-distributed TOEs, the evaluator shall ensure the TSS lists the keys the Security Administrator is able to manage to include the options available (e.g. generating keys, importing keys, modifying keys or deleting keys) and how that how those operations are performed.

Section 6.1, Table 19 (FMT_MTD.1/CryptoKeys) in [ST] explains that the TOE provides the ability for authorized administrators to access TOE data, such as audit data, configuration data, security attributes (such as cryptographic keys and certificates used in VPN), routing tables, and session thresholds. This access takes the form of management activity that will create keys, import keys, configuring the use of keys and destroy (zeroize) keys. The keys that can be managed are associated with a CSR (see FIA_X509_EXT.3/Rev in this table), x509v3 certificates, and SSH public keys.

Other rows in Table 19 provide the following:

FIA_X509_EXT.3/Rev indicates that the administrator can generate RSA or ECDSA key pairs on the TOE that can be embedded in a CSR created by the TOE. The administrator can then manually export the CSR to an external CA and then once a certificate has been issued, can import that certificate into the TOE.

FCS_SSHS_EXT.1 states that the TOE's implementation of SSHv2 supports public key algorithm RSA for signing and verification and also supports public key based authentication for administrators.

FCS_IPSEC_EXT.1 indicates that pre-shared keys (text-based or bit-based) can be configured by an administrator on the TOE for IPsec connection authentication.

FCS_CKM.4 provides commands that can be triggered manually by an administrator to delete/zeroize RSA and ECDSA keys.

Component Guidance Assurance Activities: For distributed TOEs see chapter 2.4.1.2.

For non-distributed TOEs, the evaluator shall also ensure the Guidance Documentation lists the keys the Security Administrator is able to manage to include the options available (e.g. generating keys, importing keys, modifying keys or deleting keys) and how that how those operations are performed.

The “ECDSA Key Generation” section in [AdminGuide] describes how to generate and zeroize a secret ECDSA key for used by the ASA SSH Server.

The section entitled "Managing Public Key Infrastructure (PKI) Keys" in [AdminGuide] describes how to generate and zeroize secret RSA or ECDSA keys associated with identity certificates that will be used by the ASA in TLS or IPsec protocol negotiations.

The section entitled " Post Quantum Preshared Key (PPK) Configuration" in [AdminGuide] describes how to import a *post-quantum-key* used to authenticate an IPsec peer.

Component Testing Assurance Activities: The evaluator shall try to perform at least one of the related actions (modify, delete, generate/import) without prior authentication as security administrator (either by authentication as a non-administrative user, if supported, or without authentication at all). Attempts to perform related actions without prior authentication should fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt to manage cryptographic keys can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator. The evaluator shall try to perform at least one of the related actions with prior authentication as security administrator. This attempt should be successful.

The test NDCPP22e:FIA_UIA_EXT.1-t3 demonstrates that there are no functions available to users prior to login. Testing activities for NDcPP22e:FIA_X509_EXT.3 demonstrates a successful attempt of the related cryptographic key manipulation actions by a prior authenticated security administrator.

The evaluator also provided several screenshots of testing activities reinforcing this. Attempts to modify the cryptographic keys without prior authentication were unsuccessful on all interfaces. Once the evaluator had authenticated, they were able to modify the cryptographic keys.

2.6.4 Management of TSF Data (VPNGW13:FMT_MTD.1/CryptoKeys)

2.6.4.1 VPNGW13:FMT_MTD.1.1/CryptoKeys

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: There is no change to the Evaluation Activities specified for this SFR in the NDcPP Supporting Document. The PP-Module modifies this SFR to make it mandatory because of the TOE's required support for IPsec.

There is no change to the evaluation activities specified for this SFR in the NDcPP Supporting document. Please see NDcPP22e:FMT_MTD.1/CryptoKeys which demonstrates that VPN cryptographic data was included in the evaluation of this SFR.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.6.5 Specification of Management Functions - per TD0631 (NDcPP22e:FMT_SMF.1)

2.6.5.1 NDcPP22e:FMT_SMF.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The security management functions for FMT_SMF.1 are distributed throughout the cPP and are included as part of the requirements in FTA_SSL_EXT.1, FTA_SSL.3, FTA_TAB.1, FMT_MOF.1(1)/ManualUpdate, FMT_MOF.1(4)/AutoUpdate (if included in the ST), FIA_AFL.1, FIA_X509_EXT.2.2 (if included in the ST), FPT_TUD_EXT.1.2 & FPT_TUD_EXT.2.2 (if included in the ST and if they include an administrator-configurable action), FMT_MOF.1(2)/Services, and FMT_MOF.1(3)/Functions (for all of these SFRs that are included in the ST), FMT_MTD, FPT_TST_EXT, and any cryptographic management functions specified in the reference standards. Compliance to these requirements satisfies compliance with FMT_SMF.1.

(containing also requirements on Guidance Documentation and Tests)

The evaluator shall examine the TSS, Guidance Documentation and the TOE as observed during all other testing and shall confirm that the management functions specified in FMT_SMF.1 are provided by the TOE. The evaluator shall confirm that the TSS details which security management functions are available through which interface(s) (local administration interface, remote administration interface).

The evaluator shall examine the TSS and Guidance Documentation to verify they both describe the local administrative interface. The evaluator shall ensure the Guidance Documentation includes appropriate warnings for the administrator to ensure the interface is local.

For distributed TOEs with the option 'ability to configure the interaction between TOE components' the evaluator shall examine that the ways to configure the interaction between TOE components is detailed in the TSS and Guidance Documentation. The evaluator shall check that the TOE behaviour observed during testing of the configured SFRs is as described in the TSS and Guidance Documentation.

Section 6.1, Table 19 of [ST] provides the TSS which details all TOE security management functions available through the TOE interfaces. The TOE is compliant with all requirements in [ST] as identified in the TSS and Guidance assurance activities throughout this report.

Section 6.1, Table 19 (FMT_SMF.1) in [ST] states that the management functions specified in FMT_SMF.1, FMT_SMF.1/VPN[VPN] and FMT_SMF.1/FFW[FW] are available to the Security administrators through all three administrative interfaces.

Section 6.1, Table 19 (FIA_UIA_EXT.1) in [ST] describes the TOE's administrative interfaces including the local console.

Section "Administrative Access" in [AdminGuide] indicates that the console (serial port) interface is directly connected to the security appliance which is sufficient to inform the administrator how to ensure that the interface is local. Section "Authentication to the ASA" provides commands to configure authentication at each administrative interface including the serial console. Section "Password Policies" warns the administrator that if

their password expires, they will not be able to login remotely, but can still login via the local serial console to reset their password.

The TOE is not distributed.

Component Guidance Assurance Activities: See TSS Assurance Activities

The TOE is compliant with all requirements in [ST] as identified in this report.

Component Testing Assurance Activities: The evaluator tests management functions as part of testing the SFRs identified in section 2.4.4. No separate testing for FMT_SMF.1 is required unless one of the management functions in FMT_SMF.1.1 has not already been exercised under any other SFR.

All TOE security functions are identified in the guidance documentation and have been tested as documented throughout this AAR.

2.6.6 Specification of Management Functions (STFFW14E:FMT_SMF.1/FFW)

2.6.6.1 STFFW14E:FMT_SMF.1.1/FFW

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS, Guidance Documentation and the TOE as observed during all other testing and shall confirm that the management functions specified in FMT_SMF.1 are provided by the TOE. The evaluator shall confirm that the TSS details which security management functions are available through which interface(s) (local administration interface, remote administration interface).

The evaluator shall examine the TSS and Guidance Documentation to verify they both describe the local administrative interface. The evaluator shall ensure the Guidance Documentation includes appropriate warnings for the administrator to ensure the interface is local.

For distributed TOEs with the option 'ability to configure the interaction between TOE components' the evaluator shall examine that the ways to configure the interaction between TOE components is detailed in the TSS and Guidance Documentation. The evaluator shall check that the TOE behaviour observed during testing of the configured SFRs is as described in the TSS and Guidance Documentation.

See NDcPP22e:FMT_SMF.1 where the evaluation activities described were also applied to the management functions defined in the FW module.

Component Guidance Assurance Activities: See TSS Activity.

See TSS Activity.

Component Testing Assurance Activities: The evaluator tests management functions as part of testing the SFRs identified in section 2.4.4. No separate testing for FMT_SMF.1 is required unless one of the management functions in FMT_SMF.1.1 has not already been exercised under any other SFR.

The management functions identified by STFFW14e:FMT_SMF.1 were used (at least) during the identified tests.

- Ability to configure firewall rules (STFFW14e:FFW_RUL_EXT.1 and VPNGW11:FPF_RUL_EXT.1)

2.6.7 Specification of Management Functions (VPNGW13:FMT_SMF.1/VPN)

2.6.7.1 VPNGW13:FMT_SMF.1.1/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to confirm that all management functions specified in FMT_SMF.1/VPN are provided by the TOE. As with FMT_SMF.1 in the Base-PP, the evaluator shall ensure that the TSS identifies what logical interfaces are used to perform these functions and that this includes a description of the local administrative interface.

See NDcPP22e:FMT_SMF.1. There is no change to the Evaluation Activities specified for this SFR in the NDcPP Supporting document.

Component Guidance Assurance Activities: The evaluator shall examine the operational guidance to confirm that all management functions specified in FMT_SMF.1/VPN are provided by the TOE. As with FMT_SMF.1 in the Base-PP, the evaluator shall ensure that the operational guidance identifies what logical interfaces are used to perform these functions and that this includes a description of the local administrative interface.

The section entitled "Configure Extended ACLs" in [AdminGuide] describes how an access-list can be created to define a set of rules to constrain traffic flows.

The “Access Lists” section of the [AdminGuide] states that the access-list command operates on a first-match basis. Therefore, the last rule added to the access list is the last rule checked. Administrators must take note of this when entering the initial rules during the configuration, as it may impact the remainder of the rule parsing.

Finally, the section entitled "Interface Types" in [AdminGuide] describes the commands that allow a named set of access list entries to be grouped and associated with a specific interface.

Refer to the Guidance assurance activities in NDcPP22e:FIA_UIA_EXT.1.2 which describe all three administrative interfaces.

Component Testing Assurance Activities: The evaluator tests management functions as part of performing other test EAs. No separate testing for FMT_SMF.1/VPN is required unless one of the management functions in FMT_SMF.1.1/VPN has not already been exercised under any other SFR.

All TOE security functions are identified in the administration guide and have been tested as documented throughout this AAR.

2.6.8 Restrictions on Security Roles (NDcPP22e:FMT_SMR.2)

2.6.8.1 NDcPP22e:FMT_SMR.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.8.2 NDcPP22e:FMT_SMR.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.8.3 NDcPP22e:FMT_SMR.2.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it details the TOE supported roles and any restrictions of the roles involving administration of the TOE.

Section 6.1, Table 19 in [ST] provides the following:

FMT_SMR.2 states that the TOE supports multiple levels of administrators, the highest of which is a privilege 15. In this evaluation, privilege 15 would be the equivalent of the authorized administrator with full read-write access. Multiple level 15 administrators with individual usernames can be created. Usernames defined within the local user database are distinguished based on their privilege level (0-15) and the service-type attribute assigned to the username, which by default is “admin”, allowing the username to authenticate (with valid password) to admin interfaces. The TOE also supports creating of VPN User accounts, which cannot login locally to the TOE, but can only authenticate VPN sessions initiated from VPN Clients. VPN users are accounts with privilege level 0, and/or with their service-type attribute set to “remote-access”.

FMT_MTD.1/CoreData states that each predefined and administratively configured privilege level has a default set of permissions that will grant them access to the TOE data, though with some privilege levels, the access is limited. The TOE performs role-based authorization, using TOE platform authorization mechanisms, to grant access to the semi-privileged and privileged levels. For the purposes of this evaluation, the privileged level is equivalent to full administrative access to the CLI or GUI, and equivalent to privilege level 15. The term "authorized administrator" or "Security Administrator" is used in [ST] to refer to any user which has been assigned to a privilege level that is permitted to perform the relevant action.

FMT_SMF.1 states that the TOE is configured to restrict the ability to enter privileged configuration mode to level 15 users (the authorized administrator) once AAA authorizations have been enabled. Privileged configuration (EXEC) mode is where the commands are available to modify user attributes, operation of the TOE ('reload'), authentication functions, audit trail management, communication with authorized external IT entities, information flow rules, modify the timestamp, specify limits for authentication failures.

Component Guidance Assurance Activities: The evaluator shall review the guidance documentation to ensure that it contains instructions for administering the TOE both locally and remotely, including any configuration that needs to be performed on the client for remote administration.

The "Administrative Access" section of the [AdminGuide] describes the three methods by which the administrator can manage the security appliance: Console (serial port), SSH and TLS. It notes that Telnet is not permitted for management on the TOE and must remain in its default state which is disabled. The "Administration" section of the [AdminGuide] describes the security functionality that an administrator can perform via the local console, remotely via SSH or remotely via TLS (on the ASDM).

The "Enabling HTTPS Access" section of the [AdminGuide] provides instructions for configuring the HTTPS server to allow HTTPS connections to the TOE in order to access the ASDM. The "Accessing ASDM From your Workstation" section of the [AdminGuide] describes how to access the TOE via a web browser.

The "Configuring SSH" section of the [AdminGuide] provides instructions for configuring inbound SSH sessions for remote administration of the ASA.

The "Local and Remote Access" section of the [AdminGuide] describes how administrative access to any administrative interface (console and/or HTTPS/TLS) can be configured to use remote AAA (RADIUS) authentication, and/or local authentication. The "Authorized Administrator" section of the [AdminGuide] indicates that the Privilege Level 15 authorized administrator is allowed access via CLI and ASDM (HTTPS).

The "Passwords" section of the [AdminGuide] provides guidance on configuring password complexity and minimum length. The "Using Pre-Shared Keys" section of the [AdminGuide] describes how to use the preshared key authentication method and it provides guidance to the user for creating complex strong keys.

The "Login Banners" section of the [AdminGuide] provides instructions for configuring login banners for the CLI (console and SSH access) as well as for the GUI (ASDM).

Component Testing Assurance Activities: In the course of performing the testing activities for the evaluation, the evaluator shall use all supported interfaces, although it is not necessary to repeat each test involving an administrative action with each interface. The evaluator shall ensure, however, that each supported method of administering the TOE that conforms to the requirements of this cPP be tested; for instance, if the TOE can be administered through a local hardware interface; SSH; and TLS/HTTPS; then all three methods of administration must be exercised during the evaluation team's test activities.

The TOE is administered through either an HTTPS protected UI (ASDM), an SSH protected CLI, or the console CLI. All three were used for varying configuration operations. ASDM was utilized heavily during firewall testing, while the CLI (both console and SSH) was used extensively for SSH and TLS testing.

All of these interfaces were exercised during NDcPP22e:FIA_UIA_EXT.1 and the various FTA test cases.

2.7 Packet Filtering (FPF)

2.7.1 Packet Filtering Rules (VPNGW13:FPF_RUL_EXT.1)

2.7.1.1 VPNGW13:FPF_RUL_EXT.1.1

TSS Assurance Activities: The evaluator shall verify that the TSS provide a description of the TOE's initialization/startup process, which clearly indicates where processing of network packets begins to take place, and provides a discussion that supports the assertion that packets cannot flow during this process.

The evaluator shall verify that the TSS also includes a narrative that identifies the components (e.g., active entity such as a process or task) involved in processing the network packets and describes the safeguards that would prevent packets flowing through the TOE without applying the ruleset in the event of a component failure. This could include the failure of a component, such as a process being terminated, or a failure within a component, such as memory buffers full and cannot process packets.

Section 6.1, Table 19 (FPF_RUL_EXT.1[VPN]) of [ST] states that during the boot cycle, the TOE first powers on hardware, loads the image, and executes the power on self-tests. Until the power on self-tests successfully complete, the interfaces to the TOE are deactivated. Once the tests complete, the interfaces become active and the rules associated with the interface become immediately operational. There is no state during initialization/ startup that the access lists are not enforced on an interface.

The TOE enforces information flow policies on network packets that are received by TOE interfaces and leave the TOE through other TOE interfaces. When network packets are received on a TOE interface, the TOE verifies whether the network traffic is allowed or not and performs one of the following actions, pass/not pass information, as well as optional logging.

The TOE implements rules that define the permitted flow of traffic between interfaces of the TOE for unauthenticated traffic.

Packets will be dropped unless a specific rule has been set up to allow the packet to pass (where the attributes of the packet match the attributes in the rule and the action associated with the rule is to pass traffic). Rules are

enforced on a first match basis from the top down. As soon as a match is found the action associated with the rule is applied.

Section 6.1, Table 21 (FPT_FLS.1/SelfTest [VPN]) & FFW_RUL_EXT.1.1[FW], FFW_RUL_EXT.1.2[FW]) of [ST] states that whenever a failure occurs within the TOE that results in the TOE ceasing operation, the TOE securely disables its interfaces to prevent the unintentional flow of any information to or from the TOE and reloads. So long as the failures persist, the TOE will continue to reload. This functionally prevents any failure from causing an unauthorized information flow. There are no failures that circumvent this protection. Should a failure occur in a subcomponent non-critical to TOE operations but critical to one or more traffic flows, the TOE will continue to function but all traffic flow through the failed subcomponent will be dropped.

Guidance Assurance Activities: The operational guidance associated with this requirement is assessed in the subsequent test EAs.

Please refer to the subsequent test assurance activities below.

Testing Assurance Activities: Test 1: The evaluator shall attempt to get network traffic to flow through the TOE while the TOE is being initialized. A steady flow of network packets that would otherwise be denied by the ruleset should be sourced and directed to a host. The evaluator shall use a packet sniffer to verify none of the generated network traffic is permitted through the TOE during initialization.

Test 2: The evaluator shall attempt to get network traffic to flow through the TOE while the TOE is being initialized. A steady flow of network packets that would be permitted by the ruleset should be sourced and directed to a host. The evaluator shall use a packet sniffer to verify none of the generated network traffic is permitted through the TOE during initialization and is only permitted once initialization is complete.

Note: The remaining testing associated with application of the ruleset is addressed in the subsequent test Evaluation Activities.

These tests were performed and described as part of the component testing assurance activities for FFW_RUL_EXT.1.1 where the evaluator determined that none of the generated network traffic is permitted through the firewall during initialization and is only permitted once initialization is complete.

2.7.1.2 VPNGW13:FPF_RUL_EXT.1.2

TSS Assurance Activities: There are no EAs specified for this element. Definition of packet filtering policy, association of operations with packet filtering rules, and association of these rules to network interfaces is described collectively under FPF_RUL_EXT.1.4.

See FPF_RUL_EXT.1.4.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.1.3 VPNGW13:FPF_RUL_EXT.1.3

TSS Assurance Activities: There are no EAs specified for this element. Definition of packet filtering policy, association of operations with packet filtering rules, and association of these rules to network interfaces is described collectively under FPF_RUL_EXT.1.4.

See FPF_RUL_EXT.1.4.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.1.4 VPNGW13:FPF_RUL_EXT.1.4

TSS Assurance Activities: The evaluator shall verify that the TSS describes a packet filtering policy that can use the following fields for each identified protocol, and that the RFCs identified for each protocol are supported:

- IPv4 (RFC 791)

o source address

o destination address

o Protocol

- IPv6 (RFC 8200)

o source address

o destination address

o next header (protocol)

- TCP (RFC 793)

o source port

o destination port

- UDP (RFC 768)

o source port

o destination port

The evaluator shall verify that the TSS describes how conformance with the identified RFCs has been determined by the TOE developer (e.g., third party interoperability testing, protocol compliance testing).

The evaluator shall verify that each rule can identify the following actions: permit, discard, and log.

The evaluator shall verify that the TSS identifies all interface types subject to the Packet Filtering policy and explains how rules are associated with distinct network interfaces. Where interfaces can be grouped into a common interface type (e.g., where the same internal logical path is used, perhaps where a common device driver is used), they can be treated collectively as a distinct network interface.

Section 6.1, Table 19 (FPF_RUL_EXT.1 [VPN]) of [ST] states that the TOE implements rules that define the permitted flow of traffic between interfaces of the TOE for unauthenticated traffic. These rules control whether a packet is transferred from one interface to another based on:

1. Presumed address of source
2. Presumed address of destination
3. Transport layer protocol (or next header in IPv6)
4. Service used (UDP or TCP ports, both source and destination)
5. Network interface on which the connection request occurs

These rules are supported for the following protocols: RFC 791(IPv4); RFC 2460 (IPv6); RFC 793 (TCP); RFC 768 (UDP). TOE compliance with these protocols is verified via regular quality assurance, regression, and interoperability testing.

An authorized administrator can define the traffic that needs to be protected by configuring access lists (permit, deny, log) and applying these access lists to interfaces using access and crypto map sets. Therefore, traffic may be selected on the basis of the source and destination address, and optionally the Layer 4 protocol and port.

Section 6.1, Table 19 (FFW_RUL_EXT.1.3/FFW_RUL_EXT.1.4) of [ST] states that Access Control Lists (ACLs) are only enforced after they've been applied to a network interface. Any network interface can have an ACL applied to it with the "access-group" command, e.g. "access-group sample-acl in interface outside". The interface types that can be assigned to an access-group are:

- Physical interfaces
  - o Ethernet
  - o GigabitEthernet
  - o TenGigabitEthernet
  - o Management
- Port-channel interfaces (designated by a port-channel number)
- Subinterface (designated by the subinterface number)

Guidance Assurance Activities: The evaluators shall verify that the operational guidance identifies the following protocols as being supported and the following attributes as being configurable within packet filtering rules for the associated protocols:

- IPv4 (RFC 791)

o source address

o destination address

o Protocol

- IPv6 (RFC 8200)

o source address

o destination address

o next header (protocol)

- TCP (RFC 793)

o source port

o destination port

- UDP (RFC 768)

o source port

o destination port

The evaluator shall verify that the operational guidance indicates that each rule can identify the following actions: permit, discard, and log.

The evaluator shall verify that the operational guidance explains how rules are associated with distinct network interfaces.

The guidance may describe the other protocols contained within [ST] (e.g., IPsec, IKE, potentially HTTPS, SSH, and TLS) that are processed by the TOE. The evaluator shall ensure that it is made clear what protocols were not considered as part of the TOE evaluation.

The “Traffic Flow Overview” section of the [AdminGuide] states that the security appliance supports the following protocols: ICMP, ICMPv6, IPv4, IPv6, TCP and UDP. For TCP and UDP traffic, service policies operate on traffic flows, and not just individual packets. As indicated throughout this AAR, the [AdminGuide] also describes the other protocols claimed in [ST] (HTTPS, TLS, SSH, IPsec and IKE) which were considered as part of the TOE evaluation.

All traffic that goes through the ASA is inspected using the Adaptive Security Algorithm and either allowed through or dropped. The "Default Traffic Flow (without ACLs)" section of the [AdminGuide] also confirms that Access Lists are required to be set up to enable traffic to flow through the security appliance. Specific permit or deny rules are required to be applied to a protocol, a source and destination IP address or Network and optionally, the source and destination ports.

The "Configure Extended ACLs" section of the [AdminGuide] describes the 'access-list' command and how it is used to specify the actions of deny, permit or log for a rule. This section also describes how to configure the icmp code and type.

The "Overview of Traffic to be Dropped and the Related Syslog Messages" section of the [AdminGuide] includes examples of how the TOE should be configured to ensure that traffic not allowed in the TOE is dropped. This includes an indication that all of the required attributes of these protocols can be configured including source and destination ports/addresses, transport layer protocol and for ICMP, type and code.

The "Mandatory Traffic Flow Controls" section of the [AdminGuide] states that in its Common Criteria certified configuration, the ASA must be drop certain traffic at all enabled interfaces at all times. Some of the traffic that must be dropped will be dropped at all times, regardless of configuration, other traffic will be dropped by ACLs, and still other traffic will be dropped by the "ip audit" feature. The 'ip audit' feature and command usage is described in this section including how to apply the "ip audit" policies to interfaces.

The "Interface Types" section of the [AdminGuide] states that the ASA interfaces which are capable of enforcing traffic filtering (via the 'access-group' command), or terminating tunnels (e.g. via the 'crypto-map' command) are interfaces that have been "named" (via the 'nameif' command). The interface name is used in all configuration commands on the ASA instead of the interface type and ID (such as gigabitethernet0/1), and is therefore required before traffic can pass through the interface. For subinterfaces, a VLAN must be assigned to the subinterface (via the 'vlan' command) before the subinterface can be named.

The "Create an Access-List and Assigning to Crypto Map" section of the [AdminGuide] describes how to configure an access-list to define the traffic to be encrypted/decrypted, and create a crypto map that references that access-list, and defines the rest of the IPsec SA parameters. For evaluating IPsec traffic, the "crypto map *map-name* interface *interface-name*" command is used.

Testing Assurance Activities: The evaluator shall perform the following tests:

Test 1: The evaluator shall use the instructions in the operational guidance to test that packet filter rules can be created that permit, discard, and log packets for each of the following attributes:

- IPv4

o Source address

o Destination Address

o Protocol

- IPv6

o Source Address

o Destination Address

o Next Header (Protocol)

- TCP

o Source Port

o Destination Port

- UDP

o Source Port

o Destination Port

Test 2: The evaluator shall repeat Test 1 above for each distinct network interface type supported by the TOE to ensure that Packet filtering rules can be defined for each all supported types.

Note that these test activities should be performed in conjunction with those of FPF_RUL_EXT.1.6 where the effectiveness of the rules is tested; here the evaluator is just ensuring the guidance is sufficient and the TOE supports the administrator creating a ruleset based on the above attributes. The test activities for FPF_RUL_EXT.1.6 define the protocol/attribute combinations required to be tested. If those combinations are configured manually, that will fulfill the objective of these test activities, but if those combinations are configured otherwise (e.g., using automation), these test activities may be necessary in order to ensure the guidance is correct and the full range of configurations can be achieved by a TOE administrator.

Test 1 & 2: The evaluator configured firewall rules for testing of the other VPNGW11:FPF_RUL_EXT.1 (including VPNGW11:FPF_RUL_EXT.1.6) tests using instructions provided within the administrative guidance and found all necessary instructions were provided accurately. The tests performed for FPF_RUL_EXT.1.6 incorporate numerous variations of packet filtering rules that demonstrate proper enforcement of the packet filtering ruleset (e.g., permit, deny, and log rules, ICMPv*, IPv4 and IPv6, TCP and UDP, numerous ports, source & destination differences, and transport protocols).

2.7.1.5 VPNGW13:FPF_RUL_EXT.1.5

TSS Assurance Activities: The evaluator shall verify that the TSS describes the algorithm applied to incoming packets, including the processing of default rules, determination of whether a packet is part of an established session, and application of administrator defined and ordered ruleset.

Section 6.1, Table 19 of [ST] provides the following:

FPF_RUL_EXT.1 [VPN] states that the TOE implements rules that define the permitted flow of traffic between interfaces of the TOE for unauthenticated traffic. Packets will be dropped unless a specific rule has been set up to allow the packet to pass (where the attributes of the packet match the attributes in the rule and the action associated with the rule is to pass traffic). Rules are enforced on a first match basis from the top down. As soon as a match is found the action associated with the rule is applied. These rules are entered in the form of access lists at the CLI (via 'access list' and 'access group' commands).

These interfaces reject traffic when the traffic arrives on an external TOE interface, and the source address is an external IT entity on an internal network;

These interfaces reject traffic when the traffic arrives on an internal TOE interface, and the source address is an external IT entity on the external network;

These interfaces reject traffic when the traffic arrives on either an internal or external TOE interface, and the source address is an external IT entity on a broadcast network;

These interfaces reject traffic when the traffic arrives on either an internal or external TOE interface, and the source address is an external IT entity on the loopback network;

These interfaces reject requests in which the subject specifies the route for information to flow when it is in route to its destination; and

For application protocols supported by the TOE (e.g., DNS, HTTP, SMTP, and POP3), these interfaces deny any access or service requests that do not conform to its associated published protocol specification (e.g., RFC). This is accomplished through protocol filtering proxies that are designed for that purpose.

Otherwise, these interfaces pass traffic only when its source address matches the network interface originating the traffic to the network interface corresponding to the traffic's destination address.

FFW_RUL_EXT.1.1 states that the TOE allows establishment of communications between remote endpoints, and tracks the state of each session (e.g. initiating, established, and tear-down), and will clear established sessions after proper tear-down is completed as defined by each protocol, or when session timeouts are reached.

FFW_RUL_EXT.1.5 states that if it is a new connection, the TOE has to check the packet against access control lists and perform other tasks to determine if the packet is allowed or denied.

If the connection is already established, the TOE does not need to re-check packets against the ACL; matching packets can go through the "fast" path based on attributes identified in FFW_RUL_EXT.1.5.

FFW_RUL_EXT.1.6[FW] states that the TOE can be configured to implement default denial of various mal-formed packets/fragments, and other illegitimate network traffic, and can be configured to log the packets/frames that were dropped.

Guidance Assurance Activities: The evaluator shall verify that the operational guidance describes how the order of Packet filtering rules is determined and provides the necessary instructions so that an administrator can configure the order of rule processing.

The “Access Lists” section of the [AdminGuide] states that the ‘access-list’ command operates on a first-match basis. Therefore, the last rule added to the access list is the last rule checked. Administrators must take note of this when entering the initial rules during the configuration, as it may impact the remainder of the rule parsing.

Testing Assurance Activities: The evaluator shall perform the following tests:

Test 1: The evaluator shall devise two equal Packet Filtering rules with alternate operations --“ permit and discard. The rules should then be deployed in two distinct orders and in each case the evaluator shall ensure that the first rule is enforced in both cases by generating applicable packets and using packet capture and logs for confirmation.

Test 2: The evaluator shall repeat the procedure above, except that the two rules should be devised where one is a subset of the other (e.g., a specific address vs. a network segment). Again, the evaluator should test both orders to ensure that the first is enforced regardless of the specificity of the rule.

The following was performed for both IPv4 and IPv6

Test 1: The evaluator configured the TOE (according to the admin guide) with two packet filtering rules using the same matching criteria, where one rule would permit while the other would deny traffic. Packets matching the ACL entry rule were sent through the TOE and the evaluator observed that the action taken by the TOE matched the action specified by the first ACL entry.

Test 2: Continuing test 1, the evaluator repeated the procedure above, except the evaluator changed the rules to make one a subset of the other, and then tested both orders. The evaluator confirmed that the first rule is enforced regardless of the specificity of the rule.

2.7.1.6 VPNGW13:FPF_RUL_EXT.1.6

TSS Assurance Activities: The evaluator shall verify that the TSS describes the process for applying Packet Filtering rules and also that the behavior (either by default, or as configured by the administrator) is to discard packets when there is no rule match. The evaluator shall verify the TSS describes when the IPv4/IPv6 protocols supported by the TOE differ from the full list provided in the RFC Values for IPv4 and IPv6 table.

Section 6.1, Table 19 (FPF_RUL_EXT.1 [VPN]) of [ST] states that the TOE implements rules that define the permitted flow of traffic between interfaces of the TOE for unauthenticated traffic. Packets will be dropped unless a specific rule has been set up to allow the packet to pass (where the attributes of the packet match the attributes in the rule and the action associated with the rule is to pass traffic). Rules are enforced on a first match basis from the top down. As soon as a match is found the action associated with the rule is applied.

These rules are entered in the form of access lists at the CLI (via ‘access list’ and ‘access group’ commands).

These interfaces reject traffic when the traffic arrives on an external TOE interface, and the source address is an external IT entity on an internal network;

These interfaces reject traffic when the traffic arrives on an internal TOE interface, and the source address is an external IT entity on the external network;

These interfaces reject traffic when the traffic arrives on either an internal or external TOE interface, and the source address is an external IT entity on a broadcast network;

These interfaces reject traffic when the traffic arrives on either an internal or external TOE interface, and the source address is an external IT entity on the loopback network;

These interfaces reject requests in which the subject specifies the route for information to flow when it is in route to its destination; and

For application protocols supported by the TOE (e.g., DNS, HTTP, SMTP, and POP3), these interfaces deny any access or service requests that do not conform to its associated published protocol specification (e.g., RFC). This is accomplished through protocol filtering proxies that are designed for that purpose.

Otherwise, these interfaces pass traffic only when its source address matches the network interface originating the traffic to the network interface corresponding to the traffic's destination address.

The TOE supports all IPv4 protocols excluding Protocol 2 (IGMP) which is not routable and thus will not be forwarded by the TOE.

The TOE supports the following 16 IPv6 protocols:

- Transport Layer Protocol 4 - IPv4 encapsulation
- Transport Layer Protocol 6 - Transmission Control
- Transport Layer Protocol 8 - Exterior Gateway Protocol
- Transport Layer Protocol 9 - any private interior gateway
- Transport Layer Protocol 17 - User Datagram
- Transport Layer Protocol 41 - IPv6 encapsulation
- Transport Layer Protocol 46 - Reservation Protocol
- Transport Layer Protocol 47 - General Routing Encapsulation
- Transport Layer Protocol 49 - BNA
- Transport Layer Protocol 58 - ICMP for IPv6
- Transport Layer Protocol 59 - No Next Header for IPv6
- Transport Layer Protocol 88 - TCF
- Transport Layer Protocol 89 - EIGRP
- Transport Layer Protocol 105 - SCPS Transport Layer Protocol
- Transport Layer Protocol 112 - Virtual Router Redundancy Protocol
- Transport Layer Protocol 132 - Stream Control Transmission Protocol

Protocol 2 (IGMP) and Protocol 103 (PIM) are excluded as they are not routable and thus not forwarded by the TOE despite the TOE recognizing the protocol.

All other IPv6 protocols from the RFC Values for IPv4 and IPv6 table in the MOD VPNGW SD v1.3 are dropped by default by the TOE.

Guidance Assurance Activities: The evaluator shall verify that the operational guidance describes the behavior if no rules or special conditions apply to the network traffic. If the behavior is configurable, the evaluator shall verify that the operational guidance provides the appropriate instructions to configure the behavior to discard packets with no matching rules. The evaluator shall verify that the operational guidance describes the range of IPv4 and IPv6 protocols supported by the TOE.

The “Default Traffic Flow (without ACLs)” section of the [AdminGuide] states that the ASA, by default, is configured with a default DHCP address pool. The outbound interface disallows all external to internal data flows. The administrator needs to be aware of this, and ensure that the correct policy for the organization is installed and committed before users are permitted to use the security appliance. Access Lists are required to be set up to enable traffic to flow through the security appliance. Specific permit or deny rules are required to be applied to a protocol, a source and destination IP address or Network and optionally, the source and destination ports.

Table 5 and 6 provide a list the default/implicit traffic flow policy applied between the “outside” and “inside” networks when no ACLs have been applied to outside or inside interfaces of the ASA.

Section “Configure Extended ACLs” in the [AdminGuide] describes the range of IPv4 and IPv6 protocols that are supported by the TOE.

Testing Assurance Activities: The evaluator shall perform the following tests:

Test 1: The evaluator shall configure the TOE to permit and log each supported IPv4 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each supported IPv4 Transport Layer Protocol and within the configured source and destination addresses in order to ensure that the supported protocols are permitted (i.e., by capturing the packets after passing through the TOE) and logged. Any protocols not supported by the TOE must be denied.

Test 2: The evaluator shall configure the TOE to permit all traffic except to discard and log each supported IPv4 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each defined IPv4 Transport Layer Protocol and within the configured source and destination addresses in order to ensure that the supported protocols are denied (i.e., by capturing no applicable packets passing through the TOE) and logged. Any protocols not supported by the TOE must also be denied but are not required to be logged.

Test 3: The evaluator shall configure the TOE to permit and log each supported IPv4 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. Additionally, the evaluator shall configure the TOE to discard and log each supported IPv4 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with different (than those permitted above) combinations of a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each supported IPv4 Transport Layer Protocol and outside the scope of all source and destination addresses configured above in order to ensure that the supported protocols are denied (i.e., by capturing no applicable packets passing through the TOE) and logged. Any protocols not supported by the TOE must be denied.

Test 4: The evaluator shall configure the TOE to permit and log each supported IPv6 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each defined IPv6 Transport Layer Protocol and within the configured source and destination addresses in order to ensure that the supported protocols are permitted (i.e., by capturing the packets after passing through the TOE) and logged. Any protocols not supported by the TOE must be denied.

Test 5: The evaluator shall configure the TOE to permit all traffic except to discard and log each supported IPv6 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each defined IPv6 Transport Layer Protocol and within the configured source and destination addresses in order to ensure that the supported protocols are denied (i.e., by capturing no applicable packets passing through the TOE) and logged. Any protocols not supported by the TOE must also be denied but are not required to be logged.

Test 6: The evaluator shall configure the TOE to permit and log each supported IPv6 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. Additionally, the evaluator shall configure the TOE to discard and log each supported IPv6 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with different (than those permitted above) combinations of a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each defined IPv6 Transport Layer Protocol and outside the scope of all source and destination addresses configured above in order to ensure that the supported protocols are dropped (i.e., by capturing no applicable packets passing through the TOE) and logged. Any protocols not supported by the TOE must be denied.

Test 7: The evaluator shall configure the TOE to permit and log protocol 6 (TCP) using a selected source port, a selected destination port, and a selected source and destination port combination. The evaluator shall generate packets matching the configured source and destination TCP ports in order to ensure that they are permitted (i.e., by capturing the packets after passing through the TOE) and logged.

Test 8: The evaluator shall configure the TOE to discard and log protocol 6 (TCP) using a selected source port, a selected destination port, and a selected source and destination port combination. The evaluator shall generate packets matching the configured source and destination TCP ports in order to ensure that they are denied (i.e., by capturing no applicable packets passing through the TOE) and logged.

Test 9: The evaluator shall configure the TOE to permit and log protocol 17 (UDP) using a selected source port, a selected destination port, and a selected source and destination port combination. The evaluator shall generate packets matching the configured source and destination UDP ports in order to ensure that they are permitted (i.e., by capturing the packets after passing through the TOE) and logged. Here the evaluator ensures that the UDP port 500 (IKE) is included in the set of tests.

Test 10: The evaluator shall configure the TOE to discard and log protocol 17 (UDP) using a selected source port, a selected destination port, and a selected source and destination port combination. The evaluator shall generate packets matching the configured source and destination UDP ports in order to ensure that they are denied (i.e., by capturing no applicable packets passing through the TOE) and logged. Again, the evaluator ensures that UDP port 500 is included in the set of tests.

The following table identifies the RFC defined values for the protocol fields for IPv4 and IPv6 to be used in configuring and otherwise testing Packet Filtering rule definition and enforcement:

Test 1: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured. The packets that were sent were constructed such that the packet would match only one configured rule. The evaluator confirmed that packets sent using protocols that are not supported by the TOE are not passed. Additionally, supported protocols are filtered properly and the rules are enforced.

- IPv4 Protocol permitted and logged based on specific source and destination addresses.
- IPv4 Protocol permitted and logged based on specific destination addresses.
- IPv4 Protocol permitted and logged based on specific source addresses.
- IPv4 Protocol permitted and logged based on wildcard addresses.

Test 2: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured rule. The evaluator confirmed that packets sent using protocols that are not supported by the TOE are not passed. Additionally, supported protocols are filtered properly and the rules are enforced.

- IPv4 Protocol all permitted with some denied and logged based on specific source and destination addresses.
- IPv4 Protocol all permitted with some denied and logged based on specific destination addresses.
- IPv4 Protocol all permitted with some denied and logged based on specific source addresses.

- IPv4 Protocol all permitted with some denied and logged based on wildcard addresses.

Test 3: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured rule. The evaluator confirmed that packets sent using protocols that are not supported by the TOE are not passed. Additionally, supported protocols are filtered properly and the rules are enforced.

- IPv4 Protocol some permitted and logged and some denied and logged based on specific source and destination addresses.
- IPv4 Protocol some permitted and logged and some denied and logged based on specific destination addresses.
- IPv4 Protocol some permitted and logged and some denied and logged based on specific source addresses.
- IPv4 Protocol some permitted and logged and some denied and logged based on wildcard addresses.
- IPv4 Protocol some permitted and logged and some denied and logged based on default addresses.

Test 4: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured rule. The evaluator confirmed that packets sent using protocols that are not supported by the TOE are not passed. Additionally, supported protocols are filtered properly and the rules are enforced.

- IPv6 Protocol permitted and logged based on specific source and destination addresses.
- IPv6 Protocol permitted and logged based on specific destination addresses.
- IPv6 Protocol permitted and logged based on specific source addresses.
- IPv6 Protocol permitted and logged based on wildcard addresses.

Test 5: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured rule. The evaluator confirmed that packets sent using protocols that are not supported by the TOE are not passed. Additionally, supported protocols are filtered properly and the rules are enforced.

- IPv6 Protocol all permitted with some denied and logged based on specific source and destination addresses.
- IPv6 Protocol all permitted with some denied and logged based on specific destination addresses.
- IPv6 Protocol all permitted with some denied and logged based on specific source addresses.
- IPv6 Protocol all permitted with some denied and logged based on wildcard addresses.

Test 6: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured rule. The evaluator confirmed that packets sent using protocols that are not supported by the TOE are not passed. Additionally, supported protocols are filtered properly and the rules are enforced.

- IPv6 Protocol some permitted and logged and some denied and logged based on specific source and destination addresses.

- IPv6 Protocol some permitted and logged and some denied and logged based on specific destination addresses.
- IPv6 Protocol some permitted and logged and some denied and logged based on specific source addresses.
- IPv6 Protocol some permitted and logged and some denied and logged based on wildcard addresses.
- IPv6 Protocol some permitted and logged and some denied and logged based on default addresses.

Test 7: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured rule. The evaluator confirmed that packets sent to ports that are not permitted by the TOE configuration are denied. Additionally, packets sent to permitted ports are filtered properly and the rules are enforced.

- TCP (IPv4) permitted and logged based on source port.
- TCP (IPv4) permitted and logged based on destination port.
- TCP (IPv4) permitted and logged based on source and destination port.
- TCP (IPv6) permitted and logged based on source port.
- TCP (IPv6) permitted and logged based on destination port.
- TCP (IPv6) permitted and logged based on source and destination port.

Test 8: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured rule. The evaluator confirmed that packets sent to ports that are not permitted by the TOE configuration are denied. Additionally, packets sent to permitted ports are filtered properly and the rules are enforced.

- TCP (IPv4) denied and logged based on source port.
- TCP (IPv4) denied and logged based on destination port.
- TCP (IPv4) denied and logged based on source and destination port.
- TCP (IPv6) denied and logged based on source port.
- TCP (IPv6) denied and logged based on destination port.
- TCP (IPv6) denied and logged based on source and destination port.

Test 9: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured rule. The evaluator confirmed that packets sent using protocols that are not supported by the TOE are denied. Additionally, supported protocols are filtered properly and the rules are enforced.

- UDP (IPv4) permitted and logged based on source port.
- UDP (IPv4) permitted and logged based on destination port.
- UDP (IPv4) permitted and logged based on source and destination port.
- UDP (IPv6) permitted and logged based on source port.
- UDP (IPv6) permitted and logged based on destination port.
- UDP (IPv6) permitted and logged based on source and destination port.

Test 10: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured rule. The evaluator confirmed that packets sent to ports that are not permitted by the TOE configuration are denied. Additionally, packets sent to permitted ports are filtered properly and the rules are enforced.

- UDP (IPv4) denied and logged based on source port.
- UDP (IPv4) denied and logged based on destination port.
- UDP (IPv4) denied and logged based on source and destination port.
- UDP (IPv6) denied and logged based on source port.
- UDP (IPv6) denied and logged based on destination port.
- UDP (IPv6) denied and logged based on source and destination port.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.8 Protection of the TSF (FPT)

2.8.1 Protection of Administrator Passwords (NDcPP22e:FPT_APW_EXT.1)

2.8.1.1 NDcPP22e:FPT_APW_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.1.2 NDcPP22e:FPT_APW_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it details all authentication data that are subject to this requirement, and the method used to obscure the plaintext password data when stored. The TSS shall also detail passwords are stored in such a way that they are unable to be viewed through an interface designed specifically for that purpose, as outlined in the application note.

Section 6.1, Table 19 (FPT_APW_EXT.1) of [ST] states that all TOE administrative passwords are stored as PBKDF2 hashes. When an administrator sets a password via CLI or ASDM (e.g. using the "username" command via CLI), the TOE creates a PBKDF2 hash when it saves the password to the configuration. When an administrator views the

stored configuration via CLI or ASDM (e.g. using the "show running-config" command via CLI), the configuration does not display the actual password, it shows the hashed password followed by the “pbkdf2” keyword.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.8.2 Failure with Preservation of Secure State (Self-Test Failures) (VPNGW13:FPT_FLS.1/SelfTest)

2.8.2.1 VPNGW13:FPT_FLS.1.1/SelfTest

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure the TSS describes how the TOE ensures a shutdown upon a self-test failure, a failed integrity check of the TSF executable image, or a failed health test of the noise source. If there are instances when a shut-down does not occur, (e.g., a failure is deemed non- security relevant), the evaluator shall ensure that those cases are identified and a rationale is provided that supports the classification and justifies why the TOE's ability to enforce its security policies is not affected in any such instance.

Section 6.1, Table 19 in [ST] provides the following:

FPT_FLS.1/SelfTest [VPN] states that noise source health tests are run both periodically and at start-up to determine the functional health of the noise source. These tests are specifically designed to catch catastrophic losses in the overall entropy associated with the noise source. Details regarding what the noise source health tests are doing can be found in the proprietary Entropy Design document. Tests are run on the raw noise output, before the application of any conditioners. If a noise source fails the health test either at start-up or after the device is operational, the platform will be shut down.

Whenever a failure occurs within the TOE that results in the TOE ceasing operation, the TOE securely disables its interfaces to prevent the unintentional flow of any information to or from the TOE and reloads. So long as the failures persist, the TOE will continue to reload. This functionally prevents any failure from causing an unauthorized information flow. There are no failures that circumvent this protection.

FPT_TST_EXT.1 states that the TOE runs a suite of self-tests during initial start-up (power-on-self-tests or POST) to verify its correct operation. FIPS mode must be enabled in the evaluated configuration. When FIPS mode is enabled on the TOE, additional cryptographic tests and software integrity test will be run during start-up. The self-testing includes cryptographic algorithm tests (known-answer tests) that feed pre-defined data to cryptographic modules and confirm the resulting output from the modules match expected values, and firmware integrity tests that verify the digital signature of the code image using RSA-2048 with SHA-512. The cryptographic algorithm testing verifies proper operation of encryption functions, decryption functions, signature padding functions,

signature hashing functions, and random number generation. The firmware integrity testing verifies the image has not been tampered with or corrupted. If any of these self-tests fails, the TOE will cease operation.

Component Guidance Assurance Activities: The evaluator shall verify that the operational guidance provides information on the self-test failures that can cause the TOE to shut down and how to diagnose the specific failure that has occurred, including possible remediation steps if available.

Under "Self-Testing" in the "Modes of Operation" section of the [AdminGuide], it states that following operational error, the ASA reboots (once power supply is available) and enters booting mode. If there is an error during the POST test during bootup, the ASA will shut down. If any component reports failure for the POST, the system crashes and appropriate information is displayed on the screen, and saved in the crashinfo file, which can be viewed via the CLI with the 'show crashinfo console' command. Within the POST, self-tests for the cryptographic operations are performed. Tests include the cryptographic algorithm test, software integrity test and critical functions test. The guidance advises that if any of the self tests fail (e.g. the software was tampered with, or the FIPS known answer tests (KATs) did not produce the expected results) the following actions should be taken:

- If possible, review the crashinfo file. This will provide additional information on the cause of the crash
- Restart the ASA to perform POST and determine if normal operation can be resumed
- If the problem persists, contact Cisco Technical Assistance via <http://www.cisco.com/techsupport> or 1-800-553-2447
- If necessary, return the ASA to Cisco under guidance of Cisco Technical Assistance.

This section also describes DRNG Online Health Tests (OHTs) that are run in an ongoing manner (continuously), and Built-In Self Tests (BISTs) that are run at startup. These are the health tests targeted at detecting noise source failures affecting accumulation of random data. Resolution of issues in OHT and BIST tests is the same as is described above.

Component Testing Assurance Activities: None Defined

2.8.3 Protection of TSF Data (for reading of all pre-shared, symmetric and private keys) (NDcPP22e:FPT_SKP_EXT.1)

2.8.3.1 NDcPP22e:FPT_SKP_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it details how any pre-shared keys, symmetric keys, and private keys are stored and that they are unable to be viewed through an interface designed specifically for that purpose, as outlined in the application note. If these values are not stored in plaintext, the TSS shall describe how they are protected/obscured.

Section 6.1, Table 19 (FPT_SKP_EXT.1) of [ST] states that the TOE stores all private keys in a secure directory (an "opaque" virtual filesystem in flash memory called "system:") that is not accessible to administrators. Pre-shared and symmetric keys are never visible in plaintext form, they're either stored and displayed in encrypted form (AES encrypted), or are stored in a non-encrypted form and masked when showing the configuration via administrative interfaces (CLI or GUI).

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.8.4 Reliable Time Stamps - per TD0632 (NDcPP22e:FPT_STM_EXT.1)

2.8.4.1 NDcPP22e:FPT_STM_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.4.2 NDcPP22e:FPT_STM_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure that it lists each security function that makes use of time, and that it provides a description of how the time is maintained and considered reliable in the context of each of the time related functions.

If 'obtain time from the underlying virtualization system' is selected, the evaluator shall examine the TSS to ensure that it identifies the VS interface the TOE uses to obtain time. If there is a delay between updates to the time on the VS and updating the time on the TOE, the TSS shall identify the maximum possible delay.

Section 6.1, Table 19 (FPT_STM_EXT.1) of [ST] states that the TOE provides a source of date and time information for the firewall, used in audit timestamps, validating service requests, determining the validity of certificates, and for tracking time-based actions related to session management including timeouts for inactive administrative sessions (e.g., FTA_SSL_EXT.1, FTA_SSL.3 and FTA_SSL.3/VPN), and renegotiating SAs for IPsec tunnels (FCS_IPSEC_EXT.1). This function can only be accessed from within the configuration exec mode via the privileged mode of operation of the firewall. The clock function is reliant on the system clock provided by the underlying hardware. The TOE can also synchronize it’s clock with an external NTP server to receive constant time updates.

Component Guidance Assurance Activities: The evaluator examines the guidance documentation to ensure it instructs the administrator how to set the time. If the TOE supports the use of an NTP server, the guidance

documentation instructs how a communication path is established between the TOE and the NTP server, and any configuration of the NTP client on the TOE to support this communication.

If the TOE supports obtaining time from the underlying VS, the evaluator shall verify the Guidance Documentation specifies any configuration steps necessary. If no configuration is necessary, no statement is necessary in the Guidance Documentation. If there is a delay between updates to the time on the VS and updating the time on the TOE, the evaluator shall ensure the Guidance Documentation informs the administrator of the maximum possible delay.

The “Set the Date and Time” section of the [AdminGuide] provides the steps for manually configuring the date and time on the TOE. The section "Configuring Network Time Protocol (NTP)" instructs how to enable authentication with an NTP server, and configure an NTP client on the TOE.

Component Testing Assurance Activities: The evaluator shall perform the following tests:

a) Test 1: If the TOE supports direct setting of the time by the Security Administrator then the evaluator uses the guidance documentation to set the time. The evaluator shall then use an available interface to observe that the time was set correctly.

b) Test 2: If the TOE supports the use of an NTP server; the evaluator shall use the guidance documentation to configure the NTP client on the TOE, and set up a communication path with the NTP server. The evaluator will observe that the NTP server has set the time to what is expected. If the TOE supports multiple protocols for establishing a connection with the NTP server, the evaluator shall perform this test using each supported protocol claimed in the guidance documentation.

If the audit component of the TOE consists of several parts with independent time information, then the evaluator shall verify that the time information between the different parts are either synchronized or that it is possible for all audit information to relate the time information of the different part to one base information unambiguously.

c) Test 3: [conditional] If the TOE obtains time from the underlying VS, the evaluator shall record the time on the TOE, modify the time on the underlying VS, and verify the modified time is reflected by the TOE. If there is a delay between the setting the time on the VS and when the time is reflected on the TOE, the evaluator shall ensure this delay is consistent with the TSS and Guidance.

Test 1 - The evaluator followed the guidance instructions to configure the time on the TOE. The evaluator read the time from the TOE using the ‘show clock detail’ command and also found audit records confirming that the time was successfully changed.

Test 2: Refer to FCS_NTP_EXT.1.4 – Test 1 for evidence of TOE synchronization to an NTP server.

Test 3: Not applicable. The TOE does not obtain its time from the underlying virtual server, the time is configured by the Security Administrator or obtained from an NTP server.

2.8.5 TSF TESTING (NDCPP22E:FPT_TST_EXT.1)

2.8.5.1 NDcPP22e:FPT_TST_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure that it details the self-tests that are run by the TSF; this description should include an outline of what the tests are actually doing (e.g., rather than saying 'memory is tested', a description similar to 'memory is tested by writing a value to each memory location and reading it back to ensure it is identical to what was written' shall be used). The evaluator shall ensure that the TSS makes an argument that the tests are sufficient to demonstrate that the TSF is operating correctly.

For distributed TOEs the evaluator shall examine the TSS to ensure that it details which TOE component performs which self-tests and when these self-tests are run.

Section 6.1, Table 19 in [ST] provides the following:

FPT_TST_EXT.1 states that the TOE runs a suite of self-tests during initial start-up (power-on-self-tests or POST) to verify its correct operation. FIPS mode must be enabled in the evaluated configuration. When FIPS mode is enabled on the TOE, additional cryptographic tests and software integrity test will be run during start-up. The self-testing includes cryptographic algorithm tests (known-answer tests) that feed pre-defined data to cryptographic modules and confirm the resulting output from the modules match expected values, and firmware integrity tests that verify the digital signature of the code image using RSA-2048 with SHA-512. The cryptographic algorithm testing verifies proper operation of encryption functions, decryption functions, signature padding functions, signature hashing functions, and random number generation. The firmware integrity testing verifies the image has not been tampered with or corrupted. If any of these self-tests fails, the TOE will cease operation.

FPT_FLS.1/SelfTest [VPN] states that noise source health tests are run both periodically and at start-up to determine the functional health of the noise source. These tests are specifically designed to catch catastrophic losses in the overall entropy associated with the noise source. Details regarding what the noise source health tests are doing can be found in the proprietary Entropy Design document. Tests are run on the raw noise output, before the application of any conditioners. If a noise source fails the health test either at start-up or after the device is operational, the platform will be shut down. Whenever a failure occurs within the TOE that results in the TOE ceasing operation, the TOE securely disables its interfaces to prevent the unintentional flow of any information to or from the TOE and reloads. So long as the failures persist, the TOE will continue to reload. This functionally prevents any failure from causing an unauthorized information flow. There are no failures that circumvent this protection.

The evaluator contends that the tests are sufficient to ensure the correct operation of the security features because they address integrity of the TOE executing firmware and verify the correctness of the cryptographic operations (including noise source) underlying the security features described by the Security Target.

Component Guidance Assurance Activities: The evaluator shall also ensure that the guidance documentation describes the possible errors that may result from such tests, and actions the administrator should take in response; these possible errors shall correspond to those described in the TSS.

For distributed TOEs the evaluator shall ensure that the guidance documentation describes how to determine from an error message returned which TOE component has failed the self-test.

Under "Self-Testing" in the "Modes of Operation" section of the [AdminGuide], it states that following operational error, the ASA reboots (once power supply is available) and enters booting mode. If there is an error during the POST test during bootup, the ASA will shut down. If any component reports failure for the POST, the system crashes and appropriate information is displayed on the screen, and saved in the crashinfo file, which can be viewed via the CLI with the 'show crashinfo console' command. Within the POST, self-tests for the cryptographic operations are performed. Tests include the cryptographic algorithm test, software integrity test and critical functions test. The guidance advises that if any of the self tests fail (e.g. the software was tampered with, or the FIPS known answer tests (KATs) did not produce the expected results) the following actions should be taken:

- If possible, review the crashinfo file. This will provide additional information on the cause of the crash
- Restart the ASA to perform POST and determine if normal operation can be resumed
- If the problem persists, contact Cisco Technical Assistance via <http://www.cisco.com/techsupport> or 1-800-553-2447
- If necessary, return the ASA to Cisco under guidance of Cisco Technical Assistance.

In addition, the evaluators observed that this same section described the DRNG Online Health Tests that are described as being designed to measure the quality of entropy generated by the ES using both per sample and sliding window statistical tests in hardware. This also states that in the rare event that the DRNG fails during runtime, it would cease to issue random numbers rather than issue poor quality random numbers. The entropy source and extraction algorithms are designed to comply with SP800-90B.

Component Testing Assurance Activities: It is expected that at least the following tests are performed:

a) Verification of the integrity of the firmware and executable software of the TOE

b) Verification of the correct operation of the cryptographic functions necessary to fulfill any of the SFRs.

Although formal compliance is not mandated, the self-tests performed should aim for a level of confidence comparable to:

a) FIPS 140-2, chap. 4.9.1, Software/firmware integrity test for the verification of the integrity of the firmware and executable software. Note that the testing is not restricted to the cryptographic functions of the TOE.

b) FIPS 140-2, chap. 4.9.1, Cryptographic algorithm test for the verification of the correct operation of cryptographic functions. Alternatively, national requirements of any CCRA member state for the security evaluation of cryptographic functions should be considered as appropriate.

The evaluator shall either verify that the self tests described above are carried out during initial start-up or that the developer has justified any deviation from this.

For distributed TOEs the evaluator shall perform testing of self-tests on all TOE components according to the description in the TSS about which self-test are performed by which component.

During a reboot of the TOE, the evaluator confirmed that the TOE performed self-tests to verify the firmware integrity and the cryptographic functions. The output of these tests indicate that they were successful. The firmware integrity test passed and all other tests were successfully completed with no errors.

2.8.6 TSF Testing (VPNGW13:FPT_TST_EXT.1)

2.8.6.1 VPNGW13:FPT_TST_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.6.2 VPNGW13:FPT_TST_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: There is no change to the Evaluation Activities specified for this SFR in the NDcPP Supporting Document. The PP-Module requires a particular self-test to be performed, but this self-test is still evaluated using the same methods specified in the Supporting Document.

There is no change to the Evaluation Activities specified for this SFR in the NDcPP Supporting Document. See NDcPP22e:FPT_TST_EXT.1.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.8.7 Self-Test with Defined Methods (VPNGW13:FPT_TST_EXT.3)

2.8.7.1 VPNGW13:FPT_TST_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.7.2 VPNGW13:FPT_TST_EXT.3.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator verifies that the TSS describes the method used to perform self-testing on the TSF executable code, and that this method is consistent with what is described in the SFR.

Section 6.1, Table 19 (FPT_TST_EXT.1) in [ST] states that the TOE runs a suite of self-tests during initial start-up (power-on-self-tests or POST) to verify its correct operation. FIPS mode must be enabled in the evaluated configuration. When FIPS mode is enabled on the TOE, additional cryptographic tests and software integrity test will be run during start-up. The self-testing includes cryptographic algorithm tests (known-answer tests) that feed pre-defined data to cryptographic modules and confirm the resulting output from the modules match expected values, and firmware integrity tests that verify the digital signature of the code image using RSA-2048 with SHA-512. This is consistent with the cryptographic services specified in FCS_COP.1.1/SigGen.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.8.8 Trusted update (NDcPP22e:FPT_TUD_EXT.1)

2.8.8.1 NDcPP22e:FPT_TUD_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.8.2 NDcPP22e:FPT_TUD_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.8.3 NDcPP22e:FPT_TUD_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS describe how to query the currently active version. If a trusted update can be installed on the TOE with a delayed activation, the TSS needs to describe how and when the inactive version becomes active. The evaluator shall verify this description.

The evaluator shall verify that the TSS describes all TSF software update mechanisms for updating the system firmware and software (for simplicity the term 'software' will be used in the following although the requirements apply to firmware and software). The evaluator shall verify that the description includes a digital signature verification of the software before installation and that installation fails if the verification fails. Alternatively an approach using a published hash can be used. In this case the TSS shall detail this mechanism instead of the digital signature verification mechanism. The evaluator shall verify that the TSS describes the method by which the digital signature or published hash is verified to include how the candidate updates are obtained, the processing associated with verifying the digital signature or published hash of the update, and the actions that take place for both successful and unsuccessful signature verification or published hash verification.

If the options 'support automatic checking for updates' or 'support automatic updates' are chosen from the selection in FPT_TUD_EXT.1.2, the evaluator shall verify that the TSS explains what actions are involved in automatic checking or automatic updating by the TOE, respectively.

For distributed TOEs, the evaluator shall examine the TSS to ensure that it describes how all TOE components are updated, that it describes all mechanisms that support continuous proper functioning of the TOE during update (when applying updates separately to individual TOE components) and how verification of the signature or checksum is performed for each TOE component. Alternatively, this description can be provided in the guidance documentation. In that case the evaluator should examine the guidance documentation instead.

If a published hash is used to protect the trusted update mechanism, then the evaluator shall verify that the trusted update mechanism does involve an active authorization step of the Security Administrator, and that download of the published hash value, hash comparison and update is not a fully automated process involving no active authorization by the Security Administrator. In particular, authentication as Security Administration according to FMT_MOF.1/ManualUpdate needs to be part of the update process when using published hashes.

Section 6.1, Table 19 (FPT_TUD_EXT.1) of [ST] explains that the administrator can determine the current executing version of the software by querying the TOE. The administrator can determine the current executing version of the software with the CLI command “show version”. When updates are made available by Cisco, an administrator can obtain and manually install those updates.

Cisco makes updates available to customers through their website. Administrators must download the update into the TOE (a step that automatically verifies the digital signature of the image), the administrator may optionally repeat the digital signature verification using a CLI command, the administrator must issue a command to mark the image to be used on the next reload, and finally reload the TOE. The command to mark the image to be used (boot system <image>) will display the currently running TOE version and the version of the image that will boot. The new image is not running until its digital signature is deemed valid and after the reload completes.

Digital signatures are used to verify software/firmware update files (to ensure they have not been modified from the originals distributed by Cisco) before they are used to update the TOE. The update process will fail if the digital signature verification process fails. Instructions on how to perform verification and update are provided in the Admin Guide.

It also explains that the firmware integrity tests verify the digital signature of the code image using RSA 2048 with SHA 512.

Component Guidance Assurance Activities: The evaluator shall verify that the guidance documentation describes how to query the currently active version. If a trusted update can be installed on the TOE with a delayed activation, the guidance documentation needs to describe how to query the loaded but inactive version.

The evaluator shall verify that the guidance documentation describes how the verification of the authenticity of the update is performed (digital signature verification or verification of published hash). The description shall include the procedures for successful and unsuccessful verification. The description shall correspond to the description in the TSS.

If a published hash is used to protect the trusted update mechanism, the evaluator shall verify that the guidance documentation describes how the Security Administrator can obtain authentic published hash values for the updates.

For distributed TOEs the evaluator shall verify that the guidance documentation describes how the versions of individual TOE components are determined for FPT_TUD_EXT.1, how all TOE components are updated, and the error conditions that may arise from checking or applying the update (e.g. failure of signature verification, or exceeding available storage space) along with appropriate recovery actions. The guidance documentation only has to describe the procedures relevant for the Security Administrator; it does not need to give information about the internal communication that takes place when applying updates.

If this was information was not provided in the TSS: For distributed TOEs, the evaluator shall examine the Guidance Documentation to ensure that it describes how all TOE components are updated, that it describes all mechanisms that support continuous proper functioning of the TOE during update (when applying updates separately to individual TOE components) and how verification of the signature or checksum is performed for each TOE component.

If this was information was not provided in the TSS: If [ST] author indicates that a certificate-based mechanism is used for software update digital signature verification, the evaluator shall verify that the Guidance Documentation contains a description of how the certificates are contained on the device. The evaluator also ensures that the Guidance Documentation describes how the certificates are installed/updated/selected, if necessary.

The “Verification of Image and Hardware” section in the [AdminGuide] describes how to query and verify the currently active version of the TOE by using the ‘show version’ command.

The “Verification of Image and Hardware” section in the [AdminGuide] describes how to properly verify the integrity of the downloaded ASA binary image using the ‘copy’ and/or ‘verify’ command to verify the digital

signature. Once the copy and verify are complete, the administrator must use the “boot system” and “show run boot” commands to specify which image the ASA will boot during the next system reload. The “show run boot” command is used to show the loaded but inactive version. The digital signature uses 2048-bit RSA with SHA-512. If the image verification fails, the system will not boot into operational mode. If the update succeeds, the system will boot to the new image.

The “Verification of Image and Hardware” sections in the [AdminGuide] indicate that image verification is also done during the copy, verify and reload operations. If the image verification fails here, the system will not boot into operational mode. If the update succeeds, the system will boot to the new image.

Component Testing Assurance Activities: The evaluator shall perform the following tests:

a) Test 1: The evaluator performs the version verification activity to determine the current version of the product. If a trusted update can be installed on the TOE with a delayed activation, the evaluator shall also query the most recently installed version (for this test the TOE shall be in a state where these two versions match). The evaluator obtains a legitimate update using procedures described in the guidance documentation and verifies that it is successfully installed on the TOE. For some TOEs loading the update onto the TOE and activation of the update are separate steps ('activation' could be performed e.g. by a distinct activation step or by rebooting the device). In that case the evaluator verifies after loading the update onto the TOE but before activation of the update that the current version of the product did not change but the most recently installed version has changed to the new product version. After the update, the evaluator performs the version verification activity again to verify the version correctly corresponds to that of the update and that current version of the product and most recently installed version match again.

b) Test 2 [conditional]: If the TOE itself verifies a digital signature to authorize the installation of an image to update the TOE the following test shall be performed (otherwise the test shall be omitted). The evaluator first confirms that no updates are pending and then performs the version verification activity to determine the current version of the product, verifying that it is different from the version claimed in the update(s) to be used in this test. The evaluator obtains or produces illegitimate updates as defined below, and attempts to install them on the TOE. The evaluator verifies that the TOE rejects all of the illegitimate updates. The evaluator performs this test using all of the following forms of illegitimate updates:

1) A modified version (e.g. using a hex editor) of a legitimately signed update

2) An image that has not been signed

3) An image signed with an invalid signature (e.g. by using a different key as expected for creating the signature or by manual modification of a legitimate signature)

4) If the TOE allows a delayed activation of updates the TOE must be able to display both the currently executing version and most recently installed version. The handling of version information of the most recently installed version might differ between different TOEs depending on the point in time when an attempted update is rejected. The evaluator shall verify that the TOE handles the most recently installed version information for that case as described in the guidance documentation. After the TOE has rejected the update the evaluator shall verify, that

both, current version and most recently installed version, reflect the same version information as prior to the update attempt.

c) Test 3 [conditional]: If the TOE itself verifies a hash value over an image against a published hash value (i.e. reference value) that has been imported to the TOE from outside such that the TOE itself authorizes the installation of an image to update the TOE, the following test shall be performed (otherwise the test shall be omitted). If the published hash is provided to the TOE by the Security Administrator and the verification of the hash value over the update file(s) against the published hash is performed by the TOE, then the evaluator shall perform the following tests. The evaluator first confirms that no update is pending and then performs the version verification activity to determine the current version of the product, verifying that it is different from the version claimed in the update(s) to be used in this test.

1) The evaluator obtains or produces an illegitimate update such that the hash of the update does not match the published hash. The evaluator provides the published hash value to the TOE and calculates the hash of the update either on the TOE itself (if that functionality is provided by the TOE), or else outside the TOE. The evaluator confirms that the hash values are different, and attempts to install the update on the TOE, verifying that this fails because of the difference in hash values (and that the failure is logged). Depending on the implementation of the TOE, the TOE might not allow the Security Administrator to even attempt updating the TOE after the verification of the hash value fails. In that case the verification that the hash comparison fails is regarded as sufficient verification of the correct behaviour of the TOE.

2) The evaluator uses a legitimate update and tries to perform verification of the hash value without providing the published hash value to the TOE. The evaluator confirms that this attempt fails. Depending on the implementation of the TOE it might not be possible to attempt the verification of the hash value without providing a hash value to the TOE, e.g. if the hash value needs to be handed over to the TOE as a parameter in a command line message and the syntax check of the command prevents the execution of the command without providing a hash value. In that case the mechanism that prevents the execution of this check shall be tested accordingly, e.g. that the syntax check rejects the command without providing a hash value, and the rejection of the attempt is regarded as sufficient verification of the correct behaviour of the TOE in failing to verify the hash. The evaluator then attempts to install the update on the TOE (in spite of the unsuccessful hash verification) and confirms that this fails. Depending on the implementation of the TOE, the TOE might not allow to even attempt updating the TOE after the verification of the hash value fails. In that case the verification that the hash comparison fails is regarded as sufficient verification of the correct behaviour of the TOE.

3) If the TOE allows delayed activation of updates, the TOE must be able to display both the currently executing version and most recently installed version. The handling of version information of the most recently installed version might differ between different TOEs. Depending on the point in time when the attempted update is rejected, the most recently installed version might or might not be updated. The evaluator shall verify that the TOE handles the most recently installed version information for that case as described in the guidance documentation. After the TOE has rejected the update the evaluator shall verify, that both, current version and most recently installed version, reflect the same version information as prior to the update attempt.

If the verification of the hash value over the update file(s) against the published hash is not performed by the TOE, Test 3 shall be skipped.

The evaluator shall perform Test 1, Test 2 and Test 3 (if applicable) for all methods supported (manual updates, automatic checking for updates, automatic updates).

For distributed TOEs the evaluator shall perform Test 1, Test 2 and Test 3 (if applicable) for all TOE components.

Test 1: The evaluator verified the running and loaded images. The evaluator downloaded the update into the TOE (a step that automatically verifies the digital signature of the image), the evaluator verified the running and loaded images. Next, the evaluator issued the command to mark the image to be used on the next reload, repeated the verification of the running and loaded images, and finally reloaded the TOE. The evaluator verified the running and loaded images after the reload completed.

Test 2: The evaluator attempted to perform a TOE update using a legitimate update that was modified using a hex editor. The TOE rejected the modified update and the product version did not change. Testing for invalid updates demonstrated that invalid images are immediately discarded and there is no loaded version to show, only the running version.

The evaluator attempted to perform a TOE update using an image with the digital signature removed. The TOE rejected the modified update and the product version did not change.

The evaluator attempted to perform a TOE update using an image with the digital signature manually modified. The TOE rejected the modified update and the product version did not change.

The evaluator performed a hex differential for each of the invalid images against a valid image to verify the type of corruption was consistent with the requirements.

Test 3: Not applicable. The TOE does not use published hashes for updates.

2.8.9 Trusted Update (VPNGW13:FPT_TUD_EXT.1)

2.8.9.1 VPNGW13:FPT_TUD_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.9.2 VPNGW13:FPT_TUD_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.9.3 VPNGW13:FPT_TUD_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: There is no change to the Evaluation Activities specified for this SFR in the NDcPP Supporting Document. The PP-Module modifies this SFR to mandate that a particular selection be chosen, but this selection is part of the original definition of the SFR so no new behavior is defined by the PP-Module.

There is no change to the Evaluation Activities specified for this SFR in the NDcPP Supporting Document. See NDCPP22e:FPT_TUD_EXT.1.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.9 TOE ACCESS (FTA)

2.9.1 TSF-INITIATED TERMINATION (NDCPP22E:FTA_SSL.3)

2.9.1.1 NDCPP22E:FTA_SSL.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it details the administrative remote session termination and the related inactivity time period.

Section 6.1, Table 19 (FTA_SSL.3) in [ST] states that an administrator can configure maximum inactivity times for local console, remote SSH and remote TLS administrative sessions. When a session is inactive (i.e., no session input) for the configured period of time the TOE will terminate the session, requiring the administrator to log in again to establish a new session when needed.

Component Guidance Assurance Activities: The evaluator shall confirm that the guidance documentation includes instructions for configuring the inactivity time period for remote administrative session termination.

Section “Enable Idle-Timeouts of ASDM Sessions” in the [AdminGuide] provides instructions for configuring the inactivity time period for GUI (HTTPS) remote administrative session termination.

Section “Idle Timeouts” in the [AdminGuide] provides instructions for configuring the inactivity time period for SSH remote administrative session termination.

Component Testing Assurance Activities: For each method of remote administration, the evaluator shall perform the following test:

a) Test 1: The evaluator follows the guidance documentation to configure several different values for the inactivity time period referenced in the component. For each period configured, the evaluator establishes a remote interactive session with the TOE. The evaluator then observes that the session is terminated after the configured time period.

The evaluator followed the guidance to configure the session timeout periods for SSH CLI remote sessions and confirmed that the session was terminated after the configured time period. The inactivity time period was configured using the 'cli timeout' command for periods of 1 minute, 3 minutes and 5 minutes.

2.9.2 TSF-Initiated Termination (VPN Headend) (VPNGW13:FTA_SSL.3/VPN)

2.9.2.1 VPNGW13:FTA_SSL.3.1/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to verify that it describes the ability of the TSF to terminate an inactive VPN client session.

Section 6.1, Table 19 (FTA_SSL.3[VPN]) of [ST] states that when a remote VPN client session reaches a period of inactivity, its connection is terminated and it must re-establish the connection with new authentication to resume operation. This period of inactivity is set by the administrator using 'vpn-idle-timeout' or 'default-idle-timeout' commands in the VPN configuration.

Component Guidance Assurance Activities: The evaluator shall examine the operational guidance to verify that it provides instructions to the administrator on how to configure the time limit for termination of an active VPN client session.

The "Specifying a VPN Session Idle Timeout" section of the [AdminGuide] provides instructions for configuring the user timeout period using the 'vpn-idle-timeout' command in group policy configuration mode or in username configuration mode.

Component Testing Assurance Activities: The evaluator shall perform the following tests:

Test 1: The evaluator shall follow the steps provided in the operational guidance to set the inactivity timer for five minutes. The evaluator shall then connect a VPN client to the TOE, let it sit idle for four minutes and fifty seconds, and observe that the VPN client is still connected at this time by performing an action that would require VPN access. The evaluator shall then disconnect the client, reconnect it, wait five minutes and ten seconds, attempt the

same action, and observe that it does not succeed. The evaluator shall then verify using audit log data that the VPN client session lasted for exactly five minutes.

Test 2: The evaluator shall configure the inactivity timer to ten minutes and repeat Test 1, adjusting the waiting periods and expected audit log data accordingly.

Test 1: The evaluator configured the inactivity timer for five minutes, then connected a VPN client to the TOE and waited for four minutes and fifty seconds. The evaluator then sent a ping from the TOE to the vpn client and verified the session did not close. The evaluator then established a connection and let the idle timeout expire and verified that the TOE terminated the session as expected. The VPN client log showed that the connection was terminated after 5 minutes of inactivity. The evaluator verified using audit log data that the VPN client session lasted five minutes.

Test 2: The evaluator repeated Test 1 with a time period of 10 minutes.

2.9.3 User-initiated Termination (NDcPP22e:FTA_SSL.4)

2.9.3.1 NDcPP22e:FTA_SSL.4.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it details how the local and remote administrative sessions are terminated.

Section 6.1, Table 19 (FTA_SSL.4) in [ST] states that an administrator is able to exit out of both local and remote administrative sessions, effectively terminating the session so it cannot be re-used and will require authentication to establish a new session.

Component Guidance Assurance Activities: The evaluator shall confirm that the guidance documentation states how to terminate a local or remote interactive session.

The “Terminating Administrative Sessions / Logging Out” section in the [AdminGuide] describes how the administrator logs out of a CLI session (console or SSH) using the “logout” command from any CLI mode, or using “end” to return from any config mode to exec mode followed by “exit” to terminate the session. To log out of the GUI (ASDM), the administrator should select “Exit” from the File menu.

Component Testing Assurance Activities: For each method of remote administration, the evaluator shall perform the following tests:

a) Test 1: The evaluator initiates an interactive local session with the TOE. The evaluator then follows the guidance documentation to exit or log off the session and observes that the session has been terminated.

b) Test 2: The evaluator initiates an interactive remote session with the TOE. The evaluator then follows the guidance documentation to exit or log off the session and observes that the session has been terminated.

This test was performed in conjunction with testing in NDcPP22e:FIA_UIA_EXT.1 where a logout was performed at each TOE admin interface.

2.9.4 TSF-initiated Session Locking (NDcPP22e:FTA_SSL_EXT.1)

2.9.4.1 NDcPP22e:FTA_SSL_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it details whether local administrative session locking or termination is supported and the related inactivity time period settings.

Section 6.1, Table 19 (FTA_SSL_EXT.1) in [ST] states that an administrator can configure maximum inactivity times for local console, remote SSH and remote TLS administrative sessions. When a session is inactive (i.e., no session input) for the configured period of time the TOE will terminate the session, requiring the administrator to log in again to establish a new session when needed.

Component Guidance Assurance Activities: The evaluator shall confirm that the guidance documentation states whether local administrative session locking or termination is supported and instructions for configuring the inactivity time period.

The “Idle Timeouts” section of the [AdminGuide] provides the commands for configuring an idle timeout for a local console session to specify the duration in minutes that a local console session can remain inactive before being disconnected.

Component Testing Assurance Activities: The evaluator shall perform the following test:

a) Test 1: The evaluator follows the guidance documentation to configure several different values for the inactivity time period referenced in the component. For each period configured, the evaluator establishes a local interactive session with the TOE. The evaluator then observes that the session is either locked or terminated after the configured time period. If locking was selected from the component, the evaluator then ensures that reauthentication is needed when trying to unlock the session.

The evaluator followed the guidance to configure the idle timeout periods for the Local Console session and confirmed that the session was terminated after the configured time period. The inactivity time period was configured using the 'cli timeout’ command for periods of 1, 3, and 5 minutes.

2.9.5 Default TOE Access Banners (NDcPP22e:FTA_TAB.1)

2.9.5.1 NDcPP22e:FTA_TAB.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check the TSS to ensure that it details each administrative method of access (local and remote) available to the Security Administrator (e.g., serial port, SSH, HTTPS). The evaluator shall check the TSS to ensure that all administrative methods of access available to the Security Administrator are listed and that the TSS states that the TOE is displaying an advisory notice and a consent warning message for each administrative method of access. The advisory notice and the consent warning message might be different for different administrative methods of access, and might be configured during initial configuration (e.g. via configuration file).

Section 6.1, Table 19 (FIA_UIA_EXT.1) of [ST] states that Administrative access to the TOE is facilitated through the TOE's CLI (SSH or console), and through the GUI (ASDM). The evaluator observed during testing that these are the only administrative methods of access available to the security administrator.

Section 6.1, Table 19 (FTA_TAB.1) of [ST] states that the TOE provides administrators with the capability to configure advisory banner or warning message(s) that is displayed prior to completion of the logon process at the local console or via any remote connection (e.g., SSH or HTTPS).

Component Guidance Assurance Activities: The evaluator shall check the guidance documentation to ensure that it describes how to configure the banner message.

The "Login Banners" section of the [AdminGuide] provides instructions for configuring a login banner via the CLI (local console or SSH) and via the ASDM which will be displayed prior to establishment of the administrative user session.

Component Testing Assurance Activities: The evaluator shall also perform the following test:

a) Test 1: The evaluator follows the guidance documentation to configure a notice and consent warning message. The evaluator shall then, for each method of access specified in the TSS, establish a session with the TOE. The evaluator shall verify that the notice and consent warning message is displayed in each instance.

The evaluator configured a banner and verified that the banner was displayed appropriately for console and SSH CLI logins.

2.9.6 TOE Session Establishment (VPNGW13:FTA_TSE.1)

2.9.6.1 VPNGW13:FTA_TSE.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to verify that it describes the methods by which the TSF can deny the establishment of an otherwise valid remote VPN client session (e.g., client credential is valid, not expired, not revoked, etc.), including day, time, and IP address at a minimum.

Section 6.1, Table 19 (FTA_TSE.1[VPN]) of [ST] states that the TOE allows for creation of ACLs that restrict VPN connectivity based on the client's IP address (location). These ACLs allow customization of all of these properties to allow or deny access. In addition, the 'vpn-access-hours' command can be used to restrict access based on date and time.

Component Guidance Assurance Activities: The evaluator shall review the operational guidance to determine that it provides instructions for how to enable an access restriction that will deny VPN client session establishment for each attribute described in the TSS.

The "VPN Client Access Restriction" section of the [AdminGuide] describes how to restrict VPN client access based on IP address or time range (also known as access hours). To restrict access based on IP address(s), an ACL should be defined to deny such connection from those addresses. To restrict access based on the access hours, the administrator can use the following command: *vpn-access-hours value* {time-range-name | none}.

Component Testing Assurance Activities: The evaluator shall perform the following tests:

Test 1: The evaluator shall successfully connect a remote VPN client to the TOE and then disconnect it, noting the IP address from which the client connected. The evaluator shall follow the steps described in the operational guidance to prohibit that IP address from connecting, attempt to reconnect using the same VPN client, and observe that it is not successful.

Test 2: The evaluator shall successfully connect a remote VPN client to the TOE and then disconnect it. The evaluator shall follow the steps described in the operational guidance to prohibit the VPN client from connecting on a certain day (whether this is a day of the week or specific calendar date), attempt to reconnect using the same VPN client, and observe that it is not successful.

Test 3: The evaluator shall successfully connect a remote VPN client to the TOE and then disconnect it. The evaluator shall follow the steps described in the operational guidance to prohibit the VPN client during a range of times that includes the time period during which the test occurs, attempt to reconnect using the same VPN client, and observe that it is not successful.

Test 4: [conditional] If any other attributes are identified in FTA_TSE.1, the evaluator shall conduct a test similar to tests 1 through 3 to demonstrate the enforcement of each of these attributes. The evaluator shall demonstrate a successful remote client VPN connection, configure the TSF to deny that connection based on the attribute, and demonstrate that a subsequent connection attempt is unsuccessful.

Test 1: The evaluator connected a remote VPN client to the TOE and recorded the IP address from which the client connected, then disconnected the client. The evaluator configured the TOE to prevent that IP address from connecting, then attempted to reconnect using the same VPN client, and observed that it was not successful.

Test 2: Using the same remote VPN client, the evaluator configured the TOE to prevent that IP address from connecting on a certain day. The evaluator then attempted to reconnect using the same VPN client on the configured day, and observed that it was not successful.

Test 3: Using the same remote VPN client, the evaluator configured the TOE to prevent that IP address from connecting during a range of times that includes the current time. The evaluator then attempted to reconnect using the same VPN client, and observed that it is not successful.

Test 4: Not applicable. No other attributes are claimed by the TOE.

2.9.7 VPN Client Management (VPNGW13:FTA_VCM_EXT.1)

2.9.7.1 VPNGW13:FTA_VCM_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check the TSS to verify that it asserts the ability of the TSF to assign a private IP address to a connected VPN client.

Section 6.1, Table 19 (FTA_VCM_EXT.1[VPN]) of [ST] states that the TOE provides the option to assign the remotely connecting VPN client an internal network IP address. The 'ip-local-pool' command can be used to define the range of IP and IPv6 addresses to be available for use.

Component Guidance Assurance Activities: There are no guidance EAs for this component.

There are no operational guidance Evaluation Activities for this SFR.

Component Testing Assurance Activities: The evaluator shall connect a remote VPN client to the TOE and record its IP address as well as the internal IP address of the TOE. The evaluator shall verify that the two IP addresses belong to the same network. The evaluator shall disconnect the remote VPN client and verify that the IP address of its underlying platform is no longer part of the private network identified in the previous step.

After connecting the remote VPN client to the TOE, the evaluator checked the client's IP address and ensured that the TOE successfully assigned a private IP address to the client, afterwards checking the client's status information to verify that the two IP addresses belonged to the same network. The evaluator then disconnected the VPN client and checked the status information once again. The evaluator observed that the IP address was no longer assigned to the TOE after the VPN connection was terminated.

2.10 TRUSTED PATH/CHANNELS (FTP)

2.10.1 INTER-TSF TRUSTED CHANNEL - PER TD0639 (NDCPP22E:FTP_ITC.1)

2.10.1.1 NDCPP22E:FTP_ITC.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.10.1.2 NDCPP22E:FTP_ITC.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.10.1.3 NDCPP22E:FTP_ITC.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that, for all communications with authorized IT entities identified in the requirement, each secure communication mechanism is identified in terms of the allowed protocols for that IT entity, whether the TOE acts as a server or a client, and the method of assured identification of the non-TSF endpoint. The evaluator shall also confirm that all secure communication mechanisms are described in sufficient detail to allow the evaluator to match them to the cryptographic protocol Security Functional Requirements listed in the ST.

Section 6.1, Table 19 (FTP_ITC.1) of [ST] states that the TOE uses IPsec or TLS to protect communications between itself and remote entities for the following purposes:

- The TOE protects transmission of audit records sending syslog message to a remote audit server by transmitting the message over IPsec and/or TLS.
- Connections to authentication servers (AAA servers) can be protected via IPsec tunnels. Connections with AAA servers (via RADIUS or TACACS+) can be configured for authentication of TOE administrators.
- Connections to VPN peers can be initiated from the TOE using IPsec. In addition, the TOE can establish secure VPN tunnels with IPsec VPN clients.

This also states that TOE initiates all connections to an audit server or authentication server. The TOE can be a client or server for VPN connections.

Section 6.1, Table 19 (FIA_X509_EXT.1/Rev in [ST] identifies the TOE method of assured identification of the non-TSF-endpoints, when it states that the uses X.509v3 certificates as defined by RFC 5280 to support authentication for TLS and IPsec connections.

Component Guidance Assurance Activities: The evaluator shall confirm that the guidance documentation contains instructions for establishing the allowed protocols with each authorized IT entity, and that it contains recovery instructions should a connection be unintentionally broken.

The “Configuring IPsec”, “Securing Syslog with IPsec” and “Securing RADIUS Accounting Messages with IPsec” sections of the [AdminGuide] together contain instructions for configuring IPsec for connection with each authorized IT entity. IPsec peer to peer tunnels or IPsec VPN client tunnels can be used between the ASA and a remote host for transmission of Syslog, RADIUS and VPN client connections to the ASA for remote administration (using SSH or HTTPS).

The “Configuring TLS” and “Securing Syslog with TLS” sections of the [AdminGuide] together contain instructions for configuring TLS for connection to a remote syslog server.

The “Recovering from Syslog Host Down” section of the [AdminGuide] provides recovery instructions for when the connection with the syslog server is down, while the “Clearing Security Associations” section provides recovery instructions for when the IPsec connection is broken.

Component Testing Assurance Activities: The developer shall provide to the evaluator application layer configuration settings for all secure communication mechanisms specified by the FTP_ITC.1 requirement. This information should be sufficiently detailed to allow the evaluator to determine the application layer timeout settings for each cryptographic protocol. There is no expectation that this information must be recorded in any public-facing document or report.

The evaluator shall perform the following tests:

a) Test 1: The evaluators shall ensure that communications using each protocol with each authorized IT entity is tested during the course of the evaluation, setting up the connections as described in the guidance documentation and ensuring that communication is successful.

b) Test 2: For each protocol that the TOE can initiate as defined in the requirement, the evaluator shall follow the guidance documentation to ensure that in fact the communication channel can be initiated from the TOE.

c) Test 3: The evaluator shall ensure, for each communication channel with an authorized IT entity, the channel data is not sent in plaintext.

d) Test 4: Objective: The objective of this test is to ensure that the TOE reacts appropriately to any connection outage or interruption of the route to the external IT entities.

The evaluator shall, for each instance where the TOE acts as a client utilizing a secure communication mechanism with a distinct IT entity, physically interrupt the connection of that IT entity for the following durations: i) a

duration that exceeds the TOE's application layer timeout setting, ii) a duration shorter than the application layer timeout but of sufficient length to interrupt the network link layer.

The evaluator shall ensure that, when the physical connectivity is restored, communications are appropriately protected and no TSF data is sent in plaintext.

In the case where the TOE is able to detect when the cable is removed from the device, another physical network device (e.g. a core switch) shall be used to interrupt the connection between the TOE and the distinct IT entity. The interruption shall not be performed at the virtual node (e.g. virtual switch) and must be physical in nature.

Further assurance activities are associated with the specific protocols.

For distributed TOEs the evaluator shall perform tests on all TOE components according to the mapping of external secure channels to TOE components in the Security Target.

The developer shall provide to the evaluator application layer configuration settings for all secure communication mechanisms specified by the FTP_ITC.1 requirement. This information should be sufficiently detailed to allow the evaluator to determine the application layer timeout settings for each cryptographic protocol. There is no expectation that this information must be recorded in any public- facing document or report.

The TOE utilizes IPsec for trusted channels protecting communication with an external audit server (syslog server) and an external authentication server (TACACS+ or RADIUS). The TOE also utilizes TLS to protect communications with an external audit server (syslog server).

The evaluator established a successful TLS connection between the TOE and the external audit server (the TOE initiated the connection) and began a packet capture. Within the new TLS session, the evaluator observed encrypted application data traffic between the TOE and the test server. Next the evaluator initiated the physical disruption between the TOE and the remote audit server for roughly 50 seconds. Upon reconnection the TOE continued to use the original TLS sessions and did not establish a new TLS tunnel between the TOE and the test server. The evaluator also observed audit records appearing on the syslog server after the connection was repaired. The packet capture shows that upon reconnection the TOE does not initiate a new TLS handshake.

The evaluator repeated the above steps using a duration of 6 minutes. Upon reconnection, the evaluator observed that the TOE initiated a new TLS negotiation to establish a new TLS session between the TOE and the Gossamer test server. The evaluator also observed audit records appearing on the syslog server. The packet capture shows that upon reconnection the TOE does initiate new TLS handshake.

The evaluator established a successful TOE IPsec connection supporting communication to an external audit server and began a packet capture. The packet capture shows that the TOE initiated the connection; and Application data transferred is encrypted (i.e., not plaintext). Next the evaluator initiated the physical network disruption between the TOE and the remote audit server for roughly 50 seconds. Upon reconnection, the evaluator observed audit records appearing on the syslog server. The packet capture shows that upon reconnection the TOE does not initiate ISAKMP negotiation followed by ESP traffic. The evaluator observed that no data was transmitted unprotected.

The evaluator repeated the above steps using a duration of 6 minutes and observed that the TOE initiated a new IKE negotiation to establish a new IPsec tunnel between the TOE and the Gossamer test server. The evaluator also observed audit records appearing on the syslog server. The packet capture shows that upon reconnection the TOE does initiate new ISAKMP negotiation followed by ESP traffic.

The evaluator also used the TOE to initiate an IPsec protected communication pathway to an external authentication server. Examination of the packet capture obtained during this activity showed that the connection was protected by IPsec, the TOE initiated the connection, and all application data was transferred encrypted (i.e., not plaintext). The evaluator also performed the same physical disruption test during this test and observed that no data was transmitted unprotected.

Upon completion of these activities, the resulting transcripts and packet captures were inspected. This data showed the following:

Test 1: The TOE support for IPsec protected syslog, TACACS+ and RADIUS was demonstrated, as well as support for TLS protected syslog.

Test 2: The TOE initiated the IPsec connection for both syslog, TACACS+ and RADIUS trusted channels. The TOE also initiated a TLS connection for TLS protected syslog.

Test 3: Syslog, TACACS+ and RADIUS communication were not plaintext whether protected by IPsec or TLS.

Test 4: A physical disruption in the network resulted in an IPsec and TLS session interruption and no data was transmitted unprotected.

2.10.2 Inter-TSF Trusted Channel (VPN Communications) (VPNGW13:FTP_ITC.1/VPN)

2.10.2.1 VPNGW13:FTP_ITC.1.1/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.10.2.2 VPNGW13:FTP_ITC.1.2/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.10.2.3 VPNGW13:FTP_ITC.1.3/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The EAs specified for FTP_ITC.1 in the Supporting Document for the Base-PP shall be applied for IPsec VPN communications.

Refer to NDcPP22e:FTP_ITC.1 where these activities have been performed and applied to IPsec VPN communications.

Component Guidance Assurance Activities: The EAs specified for FTP_ITC.1 in the Supporting Document for the Base-PP shall be applied for IPsec VPN communications.

Refer to NDcPP22e:FTP_ITC.1 where these activities have been performed and applied to IPsec VPN communications.

Component Testing Assurance Activities: The EAs specified for FTP_ITC.1 in the Supporting Document for the Base-PP shall be applied for IPsec VPN communications. Additional evaluation testing for IPsec is covered in FCS_IPSEC_EXT.1.

Refer to NDcPP22e:FTP_ITC.1 where these activities have been performed and applied to IPsec VPN communications.

2.10.3 Trusted Path - per TD0639 (NDcPP22e:FTP_TRP.1/Admin)

2.10.3.1 NDcPP22e:FTP_TRP.1.1/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.10.3.2 NDcPP22e:FTP_TRP.1.2/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.10.3.3 NDcPP22e:FTP_TRP.1.3/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that the methods of remote TOE administration are indicated, along with how those communications are protected. The evaluator shall also confirm that all protocols listed in the TSS in support of TOE administration are consistent with those specified in the requirement, and are included in the requirements in the ST.

Section 6.1, Table 19 (FTP_TRP.1/Admin) of [ST] states that the TOE uses SSHv2 or HTTPS (for ASDM) to provide the trusted path (with protection from disclosure and modification) for all remote administration sessions. Optionally, the TOE also supports tunneling the ASDM and/or SSH connections in IPsec VPN tunnels (peer-to-peer, or remote VPN client). The evaluator confirmed that the protocols described are consistent with the selections in FTP_TRP.1/Admin.

Component Guidance Assurance Activities: The evaluator shall confirm that the guidance documentation contains instructions for establishing the remote administrative sessions for each supported method.

The “Configuring TLS”, “Enabling HTTPS Access” and “Configuring SSH” sections of the [AdminGuide] provide instructions for configuring remote administration via SSH and HTTPS. The “Configuring IPsec” section of the [AdminGuide] provides the instructions for configuring IPsec through which ASDM and SSH connections can be optionally tunnelled.

Component Testing Assurance Activities: The evaluator shall perform the following tests:

a) Test 1: The evaluators shall ensure that communications using each specified (in the guidance documentation) remote administration method is tested during the course of the evaluation, setting up the connections as described in the guidance documentation and ensuring that communication is successful.

b) Test 2: The evaluator shall ensure, for each communication channel, the channel data is not sent in plaintext.

Further assurance activities are associated with the specific protocols.

For distributed TOEs the evaluator shall perform tests on all TOE components according to the mapping of trusted paths to TOE components in the Security Target.

The TOE supports remote administration via an SSH protected Command Line Interface, TLS protected connection to ASDM, and IPsec protection for either a CLI or ASDM session.

The evaluator performed the following on the SSH protected CLI, HTTPS protected ASDM, SSH over IPsec and HTTPS over IPsec:

a) The evaluator initiated a packet capture of traffic between a remote administrative workstation and the TOE.
b) The evaluator connected to the TOE and performed a login using an administrator account
c) The evaluator then terminated the connection by 'Logout' and terminated the packet capture.

Upon completion of these activities, the resulting transcripts and packet captures were inspected. The evaluator confirmed that the communication using each specified remote administrative method was successful and that there was no data sent in plaintext.

3. Protection Profile SAR Assurance Activities

The following sections address assurance activities specifically defined in the claimed Protection Profile that correspond with Security Assurance Requirements.

3.1 Development (ADV)

3.1.1 Basic Functional Specification (ADV_FSP.1)

Assurance Activities: The EAs for this assurance component focus on understanding the interfaces (e.g., application programing interfaces, command line interfaces, graphical user interfaces, network interfaces) described in the AGD documentation, and possibly identified in the TOE Summary Specification (TSS) in response to the SFRs. Specific evaluator actions to be performed against this documentation are identified (where relevant) for each SFR in Section 2, and in EAs for AGD, ATE and AVA SARs in other parts of Section 5.

The EAs presented in this section address the CEM work units ADV_FSP.1-1, ADV_FSP.1-2, ADV_FSP.1-3, and ADV_FSP.1-5.

The EAs are reworded for clarity and interpret the CEM work units such that they will result in more objective and repeatable actions by the evaluator. The EAs in this SD are intended to ensure the evaluators are consistently performing equivalent actions.

The documents to be examined for this assurance component in an evaluation are therefore the Security Target, AGD documentation, and any required supplementary information required by the cPP: no additional 'functional specification' documentation is necessary to satisfy the EAs. The interfaces that need to be evaluated are also identified by reference to the EAs listed for each SFR, and are expected to be identified in the context of the Security Target, AGD documentation, and any required supplementary information defined in the cPP rather than as a separate list specifically for the purposes of CC evaluation. The direct identification of documentation requirements and their assessment as part of the EAs for each SFR also means that the tracing required in ADV_FSP.1.2D (work units ADV_FSP.1-4, ADV_FSP.1-6 and ADV_FSP.1-7) is treated as implicit and no separate mapping information is required for this element.

The evaluator shall examine the interface documentation to ensure it describes the purpose and method of use for each TSFI that is identified as being security relevant.

In this context, TSFI are deemed security relevant if they are used by the administrator to configure the TOE, or to perform other administrative functions (e.g. audit review or performing updates). Additionally, those interfaces that are identified in the ST, or guidance documentation, as adhering to the security policies (as presented in the SFRs), are also considered security relevant. The intent is that these interfaces will be adequately tested, and having an understanding of how these interfaces are used in the TOE is necessary to ensure proper test coverage is applied.

The set of TSFI that are provided as evaluation evidence are contained in the Administrative Guidance and User Guidance.

The evaluator shall check the interface documentation to ensure it identifies and describes the parameters for each TSFI that is identified as being security relevant.

The evaluator shall examine the interface documentation to develop a mapping of the interfaces to SFRs.

The evaluator uses the provided documentation and first identifies, and then examines a representative set of interfaces to perform the EAs presented in Section 2, including the EAs associated with testing of the interfaces.

It should be noted that there may be some SFRs that do not have an interface that is explicitly 'mapped' to invoke the desired functionality. For example, generating a random bit string, destroying a cryptographic key that is no longer needed, or the TSF failing to a secure state, are capabilities that may be specified in SFRs, but are not invoked by an interface.

However, if the evaluator is unable to perform some other required EA because there is insufficient design and interface information, then the evaluator is entitled to conclude that an adequate functional specification has not been provided, and hence that the verdict for the ADV_FSP.1 assurance component is a 'fail'.

The Evaluation Activities for this family focus on understanding the interfaces presented in the TSS in response to the functional requirements and the interfaces presented in the AGD documentation. No additional 'functional specification' documentation is necessary to satisfy the Evaluation Activities specified in the SD.

3.2 Guidance documents (AGD)

3.2.1 Operational User Guidance (AGD_OPE.1)

Assurance Activities: The documentation must describe the process for verifying updates to the TOE for each method selected for FPT_TUD_EXT.1.3 in the Security Target. The evaluator shall verify that this process includes the following steps (per TD0536):

The evaluator performs the CEM work units associated with the AGD_OPE.1 SAR. Specific requirements and EAs on the guidance documentation are identified (where relevant) in the individual EAs for each SFR.

In addition, the evaluator performs the EAs specified below.

The evaluator shall ensure the Operational guidance documentation is distributed to administrators and users (as appropriate) as part of the TOE, so that there is a reasonable guarantee that administrators and users are aware of the existence and role of the documentation in establishing and maintaining the evaluated configuration.

The evaluator shall ensure that the Operational guidance is provided for every Operational Environment that the product supports as claimed in the Security Target and shall adequately address all platforms claimed for the TOE in the Security Target.

The evaluator shall ensure that the Operational guidance contains instructions for configuring any cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE.

The evaluator shall ensure the Operational guidance makes it clear to an administrator which security functionality and interfaces have been assessed and tested by the EAs.

In addition the evaluator shall ensure that the following requirements are also met.

a) The guidance documentation shall contain instructions for configuring any cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE.

b) The documentation must describe the process for verifying updates to the TOE by verifying a digital signature. The evaluator shall verify that this process includes the following steps:

1) Instructions for obtaining the update itself. This should include instructions for making the update accessible to the TOE (e.g., placement in a specific directory).

2) Instructions for initiating the update process, as well as discerning whether the process was successful or unsuccessful. This includes instructions that describe at least one method of validating the hash/digital signature.

c) The TOE will likely contain security functionality that does not fall in the scope of evaluation under this cPP. The guidance documentation shall make it clear to an administrator which security functionality is covered by the Evaluation Activities.

As identified throughout this AAR, the [AdminGuide] provides instructions for configuring the TOE's cryptographic security functions. The [AdminGuide] provides instructions for configuring FIPS mode such that only the cryptographic algorithms and parameters used for the evaluated configuration are available and that it is clear that no other cryptographic engines have been evaluated or tested. There are warnings and notes throughout the [AdminGuide] regarding use of functions that are and are not allowed in the evaluated configuration. There are also specific settings identified that must be enabled or disabled in order to remain CC compliant. The process for updating the TOE is described above in NDcPP22e:FPT_TUD_EXT.1.

3.2.2 Preparative Procedures (AGD_PRE.1)

Assurance Activities: As with the operational guidance, the developer should look to the Evaluation Activities to determine the required content with respect to preparative procedures.

It is noted that specific requirements for Preparative Procedures are defined in [SD] for distributed TOEs as part of the Evaluation Activities for FCO_CPC_EXT.1 and FTP_TRP.1(2)/Join.

The evaluator performs the CEM work units associated with the AGD_PRE.1 SAR. Specific requirements and EAs on the preparative documentation are identified (and where relevant are captured in the Guidance Documentation portions of the EAs) in the individual EAs for each SFR.

Preparative procedures are distributed to administrators and users (as appropriate) as part of the TOE, so that there is a reasonable guarantee that administrators and users are aware of the existence and role of the documentation in establishing and maintaining the evaluated configuration.

In addition, the evaluator performs the EAs specified below.

The evaluator shall examine the Preparative procedures to ensure they include a description of how the administrator verifies that the operational environment can fulfil its role to support the security functionality (including the requirements of the Security Objectives for the Operational Environment specified in the Security Target).

The documentation should be in an informal style and should be written with sufficient detail and explanation that they can be understood and used by the target audience (which will typically include IT staff who have general IT experience but not necessarily experience with the TOE product itself).

The evaluator shall examine the Preparative procedures to ensure they are provided for every Operational Environment that the product supports as claimed in the Security Target and shall adequately address all platforms claimed for the TOE in the Security Target.

The evaluator shall examine the preparative procedures to ensure they include instructions to successfully install the TSF in each Operational Environment.

The evaluator shall examine the preparative procedures to ensure they include instructions to manage the security of the TSF as a product and as a component of the larger operational environment.

In addition the evaluator shall ensure that the following requirements are also met.

The preparative procedures must

a) include instructions to provide a protected administrative capability; and

b) identify TOE passwords that have default values associated with them and instructions shall be provided for how these can be changed.

The evaluation team had the following documents to use when configuring the TOE:

- Cisco ASA 9.20 on Firepower 1000 and 2100 Series Preparative Procedures & Operational User Guide for the Common Criteria Certified Configuration [**AdminGuide**]

In some instances, the document referenced general Cisco manuals which the evaluator could access via web links provided in the ASA Version 9.20 Documentation Set section. The completeness of the documentation is addressed by their use in the AA's carried out in the evaluation.

3.3 Life-cycle support (ALC)

3.3.1 Labelling of the TOE (ALC_CMC.1)

Assurance Activities: This component is targeted at identifying the TOE such that it can be distinguished from other products or versions from the same vendor and can be easily specified when being procured by an end user. A label could consist of a 'hard label' (e.g., stamped into the metal, paper label) or a 'soft label' (e.g., electronically presented when queried).

The evaluator performs the CEM work units associated with ALC_CMC.1.

When evaluating that the TOE has been provided and is labelled with a unique reference, the evaluator performs the work units as presented in the CEM.

The evaluator verified that the ST, TOE and Guidance are all labeled with the same hardware versions and software. The information is specific enough to procure the TOE and it includes hardware models and software versions. The evaluator checked the TOE software version and hardware identifiers during testing by examining the actual machines used for testing.

3.3.2 TOE CM Coverage (ALC_CMS.1)

Assurance Activities: Given the scope of the TOE and its associated evaluation evidence requirements, the evaluator performs the CEM work units associated with ALC_CMS.1.

When evaluating the developer's coverage of the TOE in their CM system, the evaluator performs the work units as presented in the CEM.

See section 3.3.1 above for an explanation of how all CM items are addressed.

3.4 Tests (ATE)

3.4.1 Independent Testing - Conformance (ATE_IND.1)

Assurance Activities: Testing is performed to confirm the functionality described in the TSS as well as the guidance documentation (includes 'evaluated configuration' instructions). The focus of the testing is to confirm that the requirements specified in Section 5.1.7 are being met. The Evaluation Activities in [SD] identify the specific testing activities necessary to verify compliance with the SFRs. The evaluator produces a test report documenting the plan for and results of testing, as well as coverage arguments focused on the platform/TOE combinations that are claiming conformance to this cPP.

The focus of the testing is to confirm that the requirements specified in the SFRs are being met. Additionally, testing is performed to confirm the functionality described in the TSS, as well as the dependencies on the Operational guidance documentation is accurate.

The evaluator performs the CEM work units associated with the ATE_IND.1 SAR. Specific testing requirements and EAs are captured for each SFR in Sections 2, 3 and 4.

The evaluator should consult Appendix B when determining the appropriate strategy for testing multiple variations or models of the TOE that may be under evaluation.

Note that additional Evaluation Activities relating to evaluator testing in the case of a distributed TOE are defined in section B.4.3.1.

The evaluator created a Detailed Test Report (DTR) to address all aspects of this requirement. The DTR discusses the test configuration, test cases, expected results, and test results. The test configuration consisted of the following TOE platforms along with supporting products.

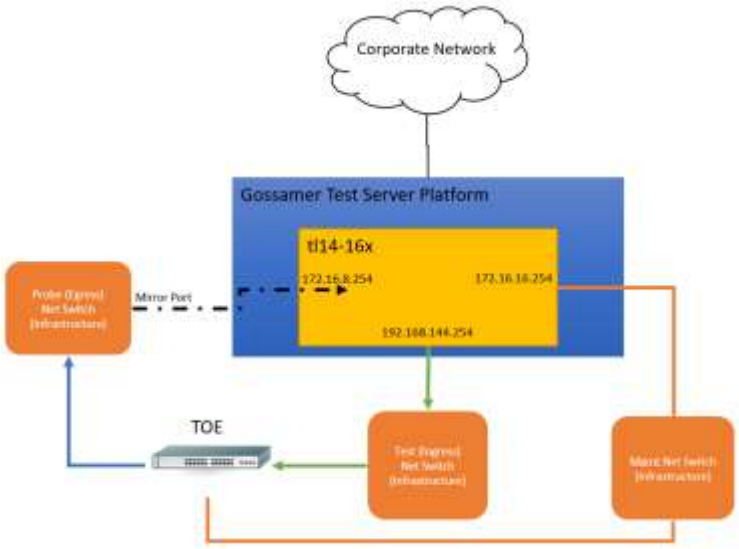

Figure 1 Test Setup

TOE Platforms:

- FP2110

Supporting Software:

The Gossamer Test servers utilized both a windows and Ubuntu environment. The Windows supporting software included the following.

- Windows 8.0, 10.0 and 11 (Evaluator Laptops)
- Wireshark version 4.0.0, 4.2.0, and 4.2.6
- Windows SSH Client - Putty – Various versions1 (between v6.8 and 8.2) were used to perform the set of SSHS tests required and to provide simple connectivity to configure the TOE through its CLI.

[1] During the course of the evaluation Gossamer has been upgrading the support tools used to perform some tests to newer versions of OpenSSH. Newer versions are necessary to support some of the stronger algorithms allowed by various Protection Profiles. The use of older version for custom tests does not compromise the actual test being performed, as the TOE must still perform properly in handling of traffic from the client.

The Gossamer Test servers with an Ubuntu environment acted as platforms to initiate testing. The test servers also acted as a syslog server.

- Openssh-client version 6.8
- Big Packet Putty, Various versions
- Openssh-client version 7.2
- Openssh-client version 8.6
- Nmap version 7.01
- Tcpdump version 4.9.3
- Libpcap version 1.7.4
- Stunnel version 5.30
- Openssl version 1.0.2g
- Strongswan version 5.3.5
- Ntpserver version 4.2.8p4
- Rsyslog version 8.16.0
- Freeradius version 3.0.15
- TACACS+ version 4.0.4.27a

3.5 Vulnerability assessment (AVA)

3.5.1 Vulnerability Survey (AVA_VAN.1)

Assurance Activities: While vulnerability analysis is inherently a subjective activity, a minimum level of analysis can be defined and some measure of objectivity and repeatability (or at least comparability) can be imposed on the vulnerability analysis process. In order to achieve such objectivity and repeatability it is important that the evaluator follows a set of well-defined activities, and documents their findings so others can follow their arguments and come to the same conclusions as the evaluator. While this does not guarantee that different evaluation facilities will identify exactly the same type of vulnerabilities or come to exactly the same conclusions, the approach defines the minimum level of analysis and the scope of that analysis, and provides Certification Bodies a measure of assurance that the minimum level of analysis is being performed by the evaluation facilities.

In order to meet these goals some refinement of the AVA_VAN.1 CEM work units is needed. The following table indicates, for each work unit in AVA_VAN.1, whether the CEM work unit is to be performed as written, or if it has been clarified by an Evaluation Activity. If clarification has been provided, a reference to this clarification is provided in the table.

Because of the level of detail required for the evaluation activities, the bulk of the instructions are contained in Appendix A, while an 'outline' of the assurance activity is provided below.

In addition to the activities specified by the CEM in accordance with Table 2, the evaluator shall perform the following activities.

The evaluator shall examine the documentation outlined below provided by the developer to confirm that it contains all required information. This documentation is in addition to the documentation already required to be supplied in response to the EAs listed previously.

The developer shall provide documentation identifying the list of software and hardware components that compose the TOE. Hardware components should identify at a minimum the processors used by the TOE. Software components include applications, the operating system and other major components that are independently identifiable and reusable (outside of the TOE), for example a web server, protocol or cryptographic libraries, (independently identifiable and reusable components are not limited to the list provided in the example). This additional documentation is merely a list of the name and version number of the components and will be used by the evaluators in formulating vulnerability hypotheses during their analysis. (TD0547 applied)

If the TOE is a distributed TOE then the developer shall provide:

a) documentation describing the allocation of requirements between distributed TOE components as in [NDcPP, 3.4]

b) a mapping of the auditable events recorded by each distributed TOE component as in [NDcPP, 6.3.3]

c) additional information in the Preparative Procedures as identified in the refinement of AGD_PRE.1 in additional information in the Preparative Procedures as identified in 3.5.1.2 and 3.6.1.2.

The evaluator formulates hypotheses in accordance with process defined in Appendix A. The evaluator documents the flaw hypotheses generated for the TOE in the report in accordance with the guidelines in Appendix A.3. The evaluator shall perform vulnerability analysis in accordance with Appendix A.2. The results of the analysis shall be documented in the report according to Appendix A.3.

The vulnerability analysis is in the Detailed Test Report (DTR) prepared by the evaluator. The vulnerability analysis includes a public search for vulnerabilities and fuzz testing. None of the public search for vulnerabilities or the fuzz testing uncovered any residual vulnerability.

The evaluation team performed a public search for vulnerabilities in order to ensure there are no publicly known and exploitable vulnerabilities in the TOE from the following sources:

- National Vulnerability Database (https://web.nvd.nist.gov/vuln/search)
    - Vulnerability Notes Database (http://www.kb.cert.org/vuls/)
    - Rapid7 Vulnerability Database (https://www.rapid7.com/db/vulnerabilities)
    - Tipping Point Zero Day Initiative (http://www.zerodayinitiative.com/advisories)
    - cve.org CVE Database (https://www.cve.org/),
    - Tenable Network Security (http://nessus.org/plugins/index.php?view=search)
    - Offensive Security Exploit Database (https://www.exploit-db.com/)

The search was performed on October 28, 2024. The search was conducted with the following search terms: "ASA", "Intel Xeon D-1526", "Intel Xeon D-1528", "Intel Xeon D-1548", "Intel Xeon D-1577", "Octeon III", "CN7230",

"CN7340", "CN7350", "CN7360", "Broadwell", "Cisco Security Crypto", "FOM", "TLS", "SSH", "IPsec", "IKE", "radius", "TACACS+".